Aalto University
School of Science
Master's Programme in ICT Innovation

Arnau Alabort

Forecasting of location occupancy by means of cell phone network data

Master's Thesis
Espoo, September 21, 2019

Supervisor: Prof. Arno Solin

Aalto University
School of Science
Master's Programme in ICT Innovation

ABSTRACT OF
MASTER'S THESIS

Author: Arnau Alabort

Title:
Forecasting of location occupancy by means of cell phone network data

Date:	September 21, 2019	**Pages:**	71
Major:	Data Science	**Code:**	SCI3095
Supervisor:	Prof. Arno Solin		

This thesis examines the feasibility of building a forecasting model capable to predict the future location occupancy for different places in the UK. For this matter, historic data of the number of people in each location as well as other data sources during the same period of time have been used.

Existing literature has researched the performance of different types of predictive models for time series. Some studies, with datasets from different sources and different time intervals, have been carried out at academia. After these studies, experts cannot agree on one best model and they tend to say that the performance of the models depends on the characteristics of the particular signal to forecast.

In this thesis, a set of time series forecasting models have been tested on different time series signals corresponding to different locations. The code implemented provides an interface for the user to select the locations, models, and several other forecasting options in order to perform a proper evaluation of the predictions. Results obtained are not biased from what could be expected from the beginning, and the outcomes obtained with the small amount of data available tend to suggest that there is no single best model for all the different predicted locations.

The use of data transformations and external regressors have also been tested in order to improve the performance of the models. Although some combinations of regressors and data transformations seem to improve some of the models, no promising conclusions can be concluded since this improvement does not apply for all the cases.

Keywords: time series, forecasting, data science, machine learning

Language: English

Contents

Chapter 1

Introduction

Time series forecasting is an important area of machine learning although it is usually neglected for the majority of Data Science and machine learning courses. By its definition, a time series is an ordered sequence of observations of a variable taken at equally spaced time intervals. Therefore, the key point about time series is that the ordering of the samples matters and this ordering imposes a certain structure on the data.

There are notable examples of time series found in our daily basis in a lot of different domains (e.g., hourly stock prices, annual company profit, quarterly house sales, hourly electricity consumption), and that is why a proper understanding of how to understand and forecast those signals is so important.

In the domain of time series we distinguish between time series analysis and time series forecasting. Time series analysis refers to the art of understanding the underlying patterns and structure producing the observed data. Some of those factors that are responsible for bringing about changes in a time series are called components (e.g., trend, seasonal, cyclical, irregular fluctuations) and can be devise from a simple analysis of the data. After having analyzed the data, the time series can be fitted with the appropriate time series forecasting models and the future values of the signal can be forecast.

Generally, time series forecasting can be classified into two main types: univariate and multivariate. A univariate time series, as the name suggests, is a series with a single time-dependent variable and therefore just one variable is used in order to predict the future values of the variable to forecast. Usually, in univariate cases the variable used is the same as the one to forecast but obviously taking its past values. On the contrary, a multivariate time series has more than one time-dependent variable and therefore the variable to forecast depends not only on its past values but also has some dependency

on other variables.

There is no clear consensus within the experts on deciding a single model as the best of all of them. Therefore, the best model can vary even if the signals to forecast have same variables providing from similar origins. Some studies have been carried out at academia with datasets from different sources and different time intervals, but none proven theories have been discovered yet. A good example of this are the MX competitions being the M3-Competition and M4-Competition the latest [23] [24].

Experts on the field tend to say that statistical old methods usually outperform machine learning models for the majority of univariate forecasting. However, as we increase the number of training samples and we start to include external regressors, this statement becomes less clear and new machine learning approaches as well as new statistical methods such as Structural Time Series (STS) models seem to perform better.

This thesis examines the feasibility of building forecasting models capable to predict the future location occupancy (number of people) for different places in the UK. For this matter, historic data of the amount of people in each location as well as other data sources during the same period of time have been used.

1.1 Research Problem and Questions

The main research problem of the thesis is to discover until what extent is possible to forecast location occupancy. To answer the research problem some research questions need to be answered. The following research questions are used in order to find a more tangible answer for the research problem.

- **Research Q1**: Is it possible to forecast future location occupancy using cell phone network data? If yes, how accurate are the forecasts obtained?
- **Research Q2**: Is there a single best model for the different locations?
- **Research Q3**: Are external variables useful in order to improve the performance of the models?

1.2 Scope of the Study

The scope of the project is to examine until what point the forecasting of location occupancy is possible using cell phone network data. This is done by

first, researching the literature about time series forecasting methods, second, coding the different methods studied fitting them with the data, and finally, properly evaluating the results providing useful information for non-technical people on the power of those forecasts.

No previous time series forecasting had been done in my work team by the time I started the project and therefore the first goal was to built a tool that was able to provide forecasts for the different models studied in any hourly data provided. The idea was to built a tool that was generic enough to be utilized in any other hourly time series data in order to be able to use the same code on similar projects inside the company with time series data.

The plan adopted to achieve that before the end of the contract was to have regular meetings with my supervisor reporting the weekly findings and advances to finally present the final results in a video conference with several people of the team which are scattered between London and Madrid.

1.3 Structure of the Study

The structure of the study can be split in three parts. First, the research about the latest time series forecasting models found in the literature comparing the old univariate methods with the latest trending models and libraries like Tensorflow Probability or Prophet.

The second part has been to prepare the data and code and fit the models. All this implementation has been wrapped in a jupyter notebook coded in Python providing a user interface where the user can select the different forecasting models and parameters. Thus, a part from the models the user can also select other features that will be explained in the Implementation section.

Finally, using the jupyter notebook implemented, the different models with the different data transformations and regressors have been analytically evaluated and compared. This evaluation has been performed for all the three locations in the dataset provided in order to find the combination of location, model, transformations and regressors performing better.

Chapter 2

Background

This section is a literature review about the research done in academia and in industry about time series analysis and time series forecasting. First, time series analysis is introduced and explained. Second, the main components appearing in the majority of time series datasets are defined. Finally, a taxonomy to classify time series forecasting models as well as a detailed description of each of the models used in this project is provided.

2.1 Time series Analysis

In the normal machine learning approach a dataset is a collection of observations that do not dependent on time. Therefore, the future is predicted but all prior observations are treated equally regardless the time when they were taken. In time series forecasting this approach is erroneous and time has to be taken into account. The time when an observation was taken is really important and it hides valuable information that will be needed to analyze and forecast the variable.

Previous time series analysis before forecasting can be very useful in time series. Even though it could not be needed for black-box machine learning approaches, when using statistical models, the models have to be provided with some information of the signal in order to perform well. The objective of a time series analysis is to spot some useful information about the signal to provide the mathematical models with. If we manage to provide a mathematical model for the sample data, we will be able to extrapolate the model in order to forecast the future values of the variable based on past experience. Hence, time series forecasting is basically taking the fit of models on historical data and using them to predict future values estimating the future from what has already happened in the past.

2.2 Components of a time series

Time series analysis provides different techniques to better understand a time series dataset. The majority of the time series are affected by similar patterns which lead to the decomposition of the time series in different components as explained by Hyndman [17] in its most famous book about time series. The three principal elements of a time series decomposition are the trend, the periodic variations, and the random fluctuations.

- **Trend**: The trend shows the general tendency of the data to increase or decrease during a long period of time. In other words, the trend is a smooth long-term average tendency of the signal.

  It is possible that tendencies may increase, decrease or are stable in different sections of time but the overall trend must be upward, downward or stable. Even though the trend is usually thought as linear, depending on the nature of the signal, it can follow non-linear patterns as well.

- **Periodic variations**: There are some components in time series which tend to repeat themselves over a certain period of time. Inside these periodic fluctuations we distinguish between seasonal variations and cyclic variations.

  The seasonal variations are the rhythmic forces which operate in a regular and periodic manner over a span of less than a year. They have the same or almost the same pattern during a period of 12 months. These variations can be caused either because by natural forces (e.g., seasons, climatic conditions) or either by man-made conventions (e.g., festivals, customs).

  The cyclical variations are the same as seasonal variations but for a not fixed period of time which tends to also be bigger. This oscillatory movement has usually a period of oscillation of more than a year.

- **Random or irregular fluctuations**:

  There is another factor which causes variations in the variable under study. They are random variations and therefore they cannot be explained neither predicted by the model.

2.3 Mathematical Models for time series

Several mathematical models have been formulated in order to model time series. Each of those models take into account different concepts which will be studied in more detail in the following section. Next, we introduce some useful notation that will be used for the rest of the document when describing each of the models

A time series consist of n values sampled at discrete times 1, 2, ..., n. In this thesis a time series of length n is represented by $\{y_t : t = 1, 2, \ldots, n\} = \{y_1, y_2, \ldots, y_n\}$. The 'hat' notation will be used to represent predictions. Hence, $\hat{y}_{t+k}$ is the forecast for the future value at time $t + k$.

To get familiar with this notation we first introduce them in two very simplistic models. As it has been stated before, many series are dominated by some of the components such as trend or seasonality. Consequently, a simple mathematical model to capture that structure will be the additive decomposition model bellow:

$$y_t = m_t + s_t + z_t, \tag{2.1}$$

where at time t, y_t is the observed series, m_t is the trend, s_t is the seasonal effect, and z_t is an error term that is, in general, a sequence of correlated random variables with mean zero.

If for example, after analyzing the time series we preview that the seasonal effect tend to increase as the trend increases, another similar model called multiplicative model would be more adequate:

$$y_t = m_t \cdot s_t + z_t. \tag{2.2}$$

2.4 Taxonomy of the Models

As discussed in previous sections different methods and different approaches can be used for time series forecasting. In this thesis we classify those methods in two different taxonomies. First, we differentiate the models between the number of variables used for the forecast, and then, we differentiate the models between the class of method used for the forecast.

When we classify the models by the number of independent variables we distinguish between univariate and multivariate models. In univariate models, just one variable is used in order to forecast the future values of the desired variable, being that variable usually the same as the one to forecast. Hence, the historic values of that variable and combinations of them are used as variables to forecast the future signal in question. In multivariate models,

a part from the historic values of the signal itself, other variables with values on the same historic time period and on the future time period that want to be forecast are used. Those other variables are often named regressors or external variables.

The other type of classification is to classify them by the type of model used to forecast. As we mentioned before, several techniques and approaches exist in order to model time series. For example, some take into account signal components such as trend or seasonality, others model the signal as a correlation with itself or other variables, and more advanced techniques, such as state space models, are able to adapt over time in order to capture the variation of forecasting parameters. From this range of various techniques, we differentiate between classic statistical methods such as Exponential Smoothing, Autoregressive (AR) or Moving Averages (MA), and other more trendy approaches like Long Short Term Memory (LSTM) neural networks or Structural Time Series models (STS) each of which will be covered in detail in the next sections.

A part from that classification, another special model is the baseline model. The baseline model is a basic model that is going to be compared with the models implemented in order to know if those are able to overcome the results of the basic one. The baseline model used in the majority of time series is the Persistence algorithm. The Persistence algorithm is a basic model that basically persists the observations for the same time in the previous season. In other words, our prediction for time t, will be $\hat{y}_t = y_{t-k}$ being k the seasonality. In this project we have built the Persistence week and Persistence year models which are persistence models with weekly and yearly seasonality respectively.

2.5 Classical Linear Forecasting Methods

The field of statistical forecasting has progressed a great deal since the early dates when Brown [13] used Exponential Smoothing, in the late 1940s, for predicting the inventory demand of items in navy shipyards. Then, later on, the introduction of the Box-Jenkins methodology to Autoregressive Integrated Moving Average (ARIMA) models [4] brought academic reputability to a field dominated until then by practitioners. This project has tested three of the most important traditional statistical methods for time series: Exponential Smoothing, Autoregressive, and Seasonal Autoregressive Integrated Moving Average with eXogenous regressors (SARIMAX) which is an improved version of the model devised by Box-Jenkins. They are explained in detail below.

2.5.1 Exponential Smoothing

Exponential Smoothing methods assign exponential decreasing weights for past observations. Higher weights are assigned to more recent observations and lower heights to further ones. For example, if we have a monthly time series, it would be reasonable to attach larger weights to observations from last month than to observations from five months ago.

In this project, a particular Exponential Smoothing algorithm named Holt-Winters [16] has been implemented. Before explaining how Holt-Winters works, first we introduce two simpler models named Simple Exponential Smoothing and Holt. Simple Exponential Smoothing is a good choice for forecasting data with no clear trend or seasonal pattern. Forecasts are calculated using weighted averages, therefore largest weights are associated with most recent observations, while smallest weights are associated with oldest observations.

$$\hat{y}_{t+1|t} = \alpha y_t + (1-\alpha)\hat{y}_{t|t-1}, \tag{2.3}$$

where $0 \leq \alpha \leq 1$ is the smoothing parameter.

The Equation 2.3 can also be expressed in a component form. Component form representations of Exponential Smoothing methods comprise a forecast equation and a smoothing equation for each of the components included in the method. In the case of Simple Exponential Smoothing, we just have one component, ℓ_t, and the equations are the following:

$$\hat{y}_{t+h|t} = \ell_t, \tag{2.4}$$

$$\ell_t = \alpha y_t + (1-\alpha)\ell_{t-1}, \tag{2.5}$$

where ℓ_t is the level (or the smoothed value) of the series at time t. This Simple Exponential Smoothing that we have presented is not particularly useful, but it will make it easier to understand when we start to use components for the Holt and Holt-Winters methods.

For every Exponential Smoothing method we need to obtain the smoothing parameters and the initial values. In particular, for Simple Exponential Smoothing, α and ℓ_0 values have to be chosen and then, once we know those values, all the forecast can be computed from the data. The unknown parameters and the initial values for any Exponential Smoothing method can be estimated by minimizing the sum of squared errors represented in Equation 2.6, which will involve a non-linear minimization problem for $t = 1, \ldots, T$, being

T the time to forecast.

$$\sum_{t=1}^{T}(y_t - \hat{y}_{t|t-1})^2 = \sum_{t=1}^{T} e_t^2. \tag{2.6}$$

The Holt method extended the Simple Exponential Smoothing to allow the forecasting of data with trend. Expressing the Holt method as a component form we obtain two smoothing equations, one for the level and one for the trend:

$$\hat{y}_{t+h|t} = \ell_t + hb_t, \tag{2.7}$$

$$\ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1}), \tag{2.8}$$

$$b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}, \tag{2.9}$$

where ℓ_t is the estimation of the level at time t, b_t denotes the estimation of the trend of the series at time t, α is the smoothing parameter for the level, $0 \leq \alpha \leq 1$, and β^* is the smoothing parameter for the trend, $0 \leq \beta^* \leq 1$.

Finally we have the Holt-Winters [16] method. This method is an extension of Holt method in order to capture seasonality. The Holt-Winters seasonal method comprises the forecast equation and three smoothing equations for the level ℓ_t, the trend b_t, and the seasonal component s_t, with corresponding smoothing parameters α, β and γ. We usually use m to denote the frequency of the seasonality. For example, for quarterly data $m = 4$, and for monthly data $m = 12$.

There are two types of Holt-Winters models differing in the nature of the seasonal component. The additive model, which models those cases when the seasonal variations are constant through the series, and the multiplicative method which is used when the seasonal variations are changing proportionally to the level of the series. The equations modelling the additive model are:

$$\hat{y}_{t+h|t} = \ell_t + hb_t + s_{t+h-m(k+1)}, \tag{2.10}$$

$$\ell_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1}), \tag{2.11}$$

$$b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}, \tag{2.12}$$

$$s_t = \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}, \tag{2.13}$$

where k is the integer part of $(h-1)/m$, which ensures that estimates of the seasonal indices used for forecasting come from the final seasonality (m) of the sample. l_t shows a weighted average between the seasonal adjusted observation ($y_t - s_{t-m}$) and the non-seasonal forecast ($l_{t-1} + b_{t-1}$) for time t. The b_t parameter is the same that we saw for the Holt linear method and the seasonal equation, s_t, shows a weighted average between the current seasonal index, ($y_t - \ell_{t-1} - b_{t-1}$), and the seasonal index for m time periods ago. The usual parameter restriction is $0 \leq \gamma \leq 1 - \alpha$, and the equations modelling the multiplicative version are:

$$\hat{y}_{t+h|t} = (\ell_t + hb_t)s_{t+h-m(k+1)}, \tag{2.14}$$

$$\ell_t = \alpha\frac{y_t}{s_{t-m}} + (1-\alpha)(\ell_{t-1} + b_{t-1}), \tag{2.15}$$

$$b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}, \tag{2.16}$$

$$s_t = \gamma\frac{y_t}{(\ell_{t-1} + b_{t-1})} + (1-\gamma)s_{t-m}. \tag{2.17}$$

Holt-Winters additive implementation of Exponential Smoothing is the one used in this thesis. Again, the idea behind Holt-Winters is to introduce level, trend, and seasonal terms that can change during the entire time period. It is out of the scope of the thesis to explain the calculus needed to find the optimum value for the different smoothing parameters estimations, but it is worth to say that the *R* and *Python* implementations usually estimate the different smoothing parameters of the model by minimizing the one-step-ahead prediction.

2.5.2 SARIMAX

There are other type of models for time series forecasting totally different from Exponential Smoothing. Exponential Smoothing is based on the trend and seasonal description of the data, while these other models focus on describing the autocorrelations of the data. In this project, a part from Exponential Smoothing, we have tested the AR and the SARIMAX models. Before introducing AR and SARIMAX, we should discuss the concept of stationary. A time series is stationary when its properties (i.e, mean, variance and autocorrelation) do not change over time. It is remarkable to say that time series with cyclic behaviour (but with no trend or seasonality) are stationary because the cycles are not of a fixed length and therefore we cannot be sure where the peaks of the cycles will be.

AR and SARIMAX requires stationary data to work properly even though, as we will explain later, SARIMAX has its own means to transform a non-stationary data to stationary. There exist some useful techniques and transformations to make time series stationary. Therefore, when we apply forecasting models that need stationary data we should first apply the transformations on the data, then fit the models with the transformed data, obtain the forecasts, and finally inverse the transformations on the forecasts to get the correct predicted values.

The main transformation used is difference. Difference is used when we want to remove the trend of the signal. Although it is possible to differentiate the data more than once, usually just one difference is sufficient. Hence, given the series y_t, after applying difference one time on the data we would obtain:

$$y'_t = y_t - y_{t-1}. \tag{2.18}$$

Some useful transformations to overcome non-constant variance would be the logarithmic, the square root, or the Box-Cox transformation. Keep in mind that if the time series have negative data we will need to add a suitable constant before applying any of these transformations. Then, after the forecasting of the signal is done, if we had added that constant we will need to subtract it after doing the inverse of the transformation on the forecasts.

Seasonality also violates stationarity due to its intrinsic autocorrealtion. To eliminate this autocorrelation we could apply the same differencing as before but with a lag equal to the seasonality instead of one:

$$y'_t = y_t - y_{t-m}, \tag{2.19}$$

being m the seasonality.

Following, we introduce three different models before explaining SARIMAX: AR, Moving Average (MA) and Autoregressive integrated Moving Average (ARIMA). All of them require stationary data but some of them already provide means to achieve that.

- **Autoregressive**

  The Autoregressive model describes the forecast output as a linear combination of its own previous values. An Autoregressive model of order p for the series y_t would satisfy the following equation:

  $$y_t = \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \cdots + \alpha_p y_{t-p} + w_t, \tag{2.20}$$

  where w_t is white noise and the α_i are the model parameters with $\alpha_p \neq 0$ for an order p process. This model is basically a multiple regression but with lagged values of y_t as predictors.

The values of the different α_i can be estimated by different procedures. The explanation of those procedures is out of the scope of this thesis but more detailed information about that can be found in Jones [20] or in Wikipedia [37]. As exposed before, AR models use to theoretically perform better with stationary data.

- **Moving Average models**:

  Moving Average model is also a regression model but rather than using past values of the forecast variable as predictors, it uses past forecast errors:

$$y_t = c + \varepsilon_t + \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} + \cdots + \theta_q\varepsilon_{t-q}, \tag{2.21}$$

  where ε_t is white noise. Concretely, this equation above reflects what we refer as an MA(q) model, or a Moving Average model of order q.

- **ARIMA**:

  One of the most commonly used techniques for modelling time series are techniques based on Autoregressive Integrated Moving Average (ARIMA) models. This model was first introduced by Box an Jenkins in 1976 [3] and it is basically a combination of three parts: an Autoregressive (AR) component, a Moving Average (MA) component, and an integrated (I) component referring to the possible requirement of an initial differencing step due to the time series showing evidence of non-stationarity. The orders of the AR, I and MA terms are commonly given by the notation (p, d, q) respectively.

  The equation defining an ARIMA model is:

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1\varepsilon_{t-1} + \cdots + \theta_q\varepsilon_{t-q} + \varepsilon_t, \tag{2.22}$$

  where y'_t is the differenced series (one or more times). The predictors include both, lagged values of y_t and lagged errors.

  Therefore, the autoregression model would be the special case of an ARIMA(p,0,0) and the Moving Average would be an ARIMA(0,0,q). Once models get more complicated it is easier to use the backshift notation in Appendix A. In backshift notation Equation 2.22 can be written as:

$$\phi_p(B)(1-B)^d y_t = c + \theta_q(B)\varepsilon_t, \tag{2.23}$$

  where $\phi_p(B)$ is equal to $(1 - \phi_1 B - \cdots - \phi_p B^p), \theta_q(B)$ is equal to $(1 + \theta_1 B + \cdots + \theta_q B^q)$, ε_t is the residual error and c is a constant.

Selecting the appropiate values for p, d and q can be difficult. However, taking a look at the autocorrelation (ACF) and partial autocorrelation (PACF) those values can be estimated in the majority of cases. The ACF plot shows the autocorrelation of the signal, measuring the relationship between y_t and y_{t-k} for different values of k. The problem when we look at the autocorrelation in the ACF is that if y_t and y_{t-1} are correlated, then y_{t-1} and y_{t-2} will be also correlated but that will be because they are both connected to y_{t-1} rather than because y_t and y_{t-2} are intrinsically correlated. To overcome this issue, we use partial correlations. Partial correlations measure the relationship between y_t and y_{t-k} after removing the effects of the lags previous to k. Nau [28] explains and summary the techniques required to select the appropriate p, d and q based on the ACF and PACF plots.

Once we have selected the order of the parameters we need to estimate the parameters of the model. We can do that using maximum likelihood estimation (MLE). This technique finds the values of the parameters which maximize the probability of obtaining the data that we have observed in our observations. Note that ARIMA models are much more complicated to estimate than regression models and different softwares will use different method of estimation and optimization algorithms.

- **SARIMAX**:

  A variant of ARIMA is the seasonal ARIMA model with exogenous variables (SARIMAX). Those are formed including seasonal terms and exogenous variables in the ARIMA model. Therefore, to the previous ARIMA model we have to include a seasonal component with AR, I and MA terms. Those terms are given by the following notation $(P, D, Q)_m$, where m is the length of the seasonality.

  A seasonal ARIMA$(p, d, q)(P, D, Q)_m$ process can be written as

$$y'_t = \sum_{i=1}^{p} \phi_i y'_{t-i} + \sum_{k=1}^{P} \Phi_k y'_{t-km} + \epsilon_t + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \sum_{g=1}^{Q} \Theta_g \epsilon_{t-gm}, \quad (2.24)$$

  with y'_t representing y_t after undergoing d and D differences. This can also be written using backshift notation as

$$\phi_p(B)\Phi_P(B^m)(1-B)^d(1-B^m)^D y_t = \theta_q(B)\Theta_Q(B^m)\varepsilon_t. \quad (2.25)$$

  Having seen Seasonal ARIMA, now we introduce external variables (regressors) to get SARIMAX models. There are two distinct approaches

in order to form SARIMAX models. The first approach is to incorporate the additional variables into the above ARIMA equation via transfer function, as described by Pankratz in [31] and Bierens in [2]. However, these approaches can lead to complicated model fitting when using more than one exogenus variable. For these reasons, some practitioners such as Hyndman in [17] built ARIMAX models as regression models with ARIMA errors. To illustrate the equations of the SARIMAX model we will use an ARMAX model for simplicity.

$$y_t = \sum_{k=1}^{b} \beta_k x_{kt} + \eta_t, \tag{2.26}$$

$$\eta_t = \sum_{i=1}^{p} \phi_i \eta_{t-i} + \epsilon_t + \sum_{j=1}^{q} \theta_j \epsilon_{t-j}, \tag{2.27}$$

where β_k are the coefficients of the b exogenous variables $x_{1t}, \ldots, x_{bt}$, and the errors of the models η_t are now modelled as an ARMA processes. In backshift notation the model can be written as:

$$y_t = \beta x_t + \frac{\theta(B)}{\phi(B)} \epsilon_t. \tag{2.28}$$

Note that for the ARIMA case , we simply replace $\phi(B)$ with $(1 - B)^d \phi(B)$.

2.5.3 SARIMAX with Fourier Terms

A drawback of SARIMAX is that just one seasonality can be used and some data have more than one seasonality modelling its shape. To overcome this inconvenience the procedure described by Skorupa [35] in a Medium post can be used. The trick applied in this blog is to utilize the exogenous variables in SARIMAX to model additional seasonalities with Fourier terms. The primarily seasonality will keep being modelled by the seasonal part of SARIMAX but the other seasonalities patterns will be modelled by Fourier terms.

Then after adding one more seasonality with Fourier terms to the SARIMAX equation saw before, we obtain the following model:

$$y_t = \sum_{k=1}^{K} [\alpha_k \sin(\frac{2\pi kt}{m}) + \beta_k \cos(\frac{2\pi kt}{m})] + \sum_{k=1}^{b} \beta_k x_{kt} + \eta_t, \tag{2.29}$$

where m represents the added seasonality and K indicates the number of Fourier terms used for that particular seasonality.

In this thesis, after analysing the ACF, PACF, and applying seasonal difference, the final ARIMA model used has been a $SARIMAX(0,0,0)(1,0,0)_{oneweek}$.

2.6 Novel Methods

A part from statistical methods we have also machine learning methods for time series modelling. Moreover, we also have included Prophet and TensorFlow Probability (TFP) which are libraries from Facebook and Google respectively that use Structural Time Series (STS) models. These STS are a family of probability models for time series that include and generalize many standard time series modeling ideas, including: autoregressive and moving average processes, local linear trends, seasonality, and regression on external variables. Those models are more interpretable than the others explained before and predictions can be interpreted by visualizing the decomposition of past data and future forecasts into structural components. Moreover, Structural time series models use a probabilistic formulation that can naturally handle missing data and provide a principled quantification of uncertainty. In order to compare the performance of this new techniques versus the old classical methodologies we have also include the implementation of these methods in the thesis.

2.6.1 Long Short Term Memory Neural Networks

Traditional neural networks [34] do not take into account previous events to predict the following and that is precisely why Recurrent Neural Networks (RNN) [36], and in particular, Long Short Term Memory (LSTM) neural networks, are the right neural network to choose in time series problems. LSTM neural networks are a type of artificial neural networks particularly designed to recognize patterns in sequences of data. The scope of this thesis does not cover the proper explanation on how an LSTM neural network operates, but if the reader is interested I truly recommend this post by Olah [30].

Before we can fit an LSTM model with time series data, we have to prepare the time series data to a data structure that can be understood by the neural network. Besides, theoretically, data also need to be stationary and observations should be scaled.

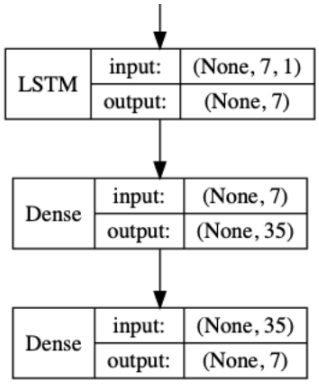


Figure 2.1: Representation of the architecture of the neural network used with LSTM models. Particularly for $n_{inputs} = 7$ and $n_{outputs} = 7$.

To transform a time series data to a proper data structure to be fit into a LSTM model, the time series has to be re-frame in a dataset ready to apply supervised learning algorithms. The correct way to do that is, starting for the beginning of the time series, to split the time series data into two adjacent non-overlapping chunks of the desired input and output neuron lengths, and then keep shifting those chunks in order to create several observations for the new dataset.

The general and simple complete neural network structure used in this thesis is the combination of an LSTM neural network fitted with n_{inputs} input neurons, followed by one fully connected layer of $n_{inputs} \cdot 5$ neurons, and followed by another fully connected of $n_{outputs}$ outputs. An example of this architecture can be seen at Figure 2.1 with the plot representation provided by the *plot_model* function of the deep learning library *keras* used in this project.

In short, our neural network will be fit with a particular number of past values of the signal n_{inputs} inputs, and the number of outputs, $n_{outputs}$, will be the equal to the number of future values that we will like to predict. In our case we will be interested in predicting the $y_{t+1}, \ldots, y_{t+n_{outputs}}$ future values of the time series just using the past values of that same signal $y_{t-n_{inputs}}, \ldots, y_{t-1}$.

For example, imagine that having the time series of Figure 2.2 representing the values taken by day of a certain measure, we want to predict the values of that measure for the following 7 days. Then, we will create the data structure represented on Figure 2.2 with the three observations under input and its three respectively outputs.

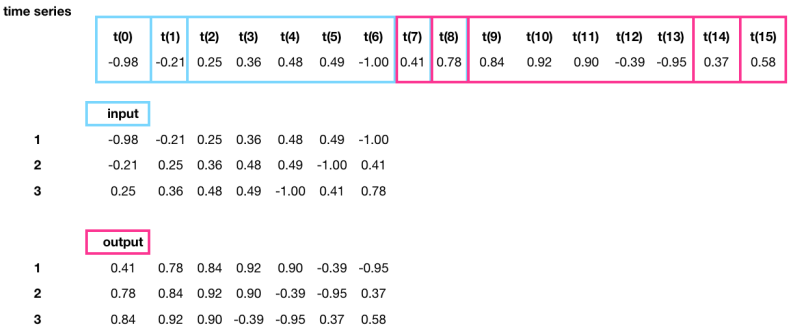


Figure 2.2: Preparation of the time series data to a data structure that can be understood by the neural network.

It is not in the scope of this thesis to find the best possible neural network architecture also because that could be different for the different analysed locations. Therefore, as we have exposed before, we have used an LSTM model architecture that seems to perform well in similar problems and consists of an LSTM neural network of n_{inputs} neurons, followed by a fully connected layer with $n_{inputs} \cdot 5$ neurons and a final fully connected layer with $n_{outputs}$ outputs. To train the network we used the mean absolute error loss and the adam optimizer with 50 epochs and a batch size of 50 as well.

2.6.2 Prophet

Prophet is an open source library published by Facebook that is based on models formed by different components [19]. It is very similar in spirit to how Bayesian Structural Time Series (BSTS) models [33] represent trend and seasonality, except that it uses Generalized Additive Models (GAM) [15] instead of a state-space representations to describe each component. Generalized Additive Models is nothing more than a fancy name for the summation of the outputs of different models. Prophet is a tool to provide time series predictions using simple intuitive parameters and has support for custom seasonalities as well as external regressors.

Until now, the approaches presented by the other models are not really intuitive for a non-skilled person in the field, and the tuning of these methods requires a thorough understanding of how the underlying time series models work. The Prophet package provides intuitive parameters which are easy to tune. Even someone who lacks expertise in forecasting models can use this tool to make meaningful predictions for a variety of problems in different

business scenarios. Prophet proposes a modular regression model with interpretable parameters that can be intuitively adjusted by analysts with domain knowledge about the time series signal. The model used by Prophet is the one proposed by Harvey and Peters [14] with three main model components: trend, seasonality, and holidays:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t, \tag{2.30}$$

where $g(t)$ is the function modelling the trend, $s(t)$ represents periodic changes (e.g., weekly, monthly, and yearly), and $h(t)$ is the component modelling events or external regressors. The ε_t term accommodates idiosyncratic changes not captured by the model which we would assume normally distributed.

The trend is modeled either as a logistic growth or as a piece-wise linear growth model for unbounded growths. In this thesis we have used the second approach and therefore our trend is modelled as follow:

$$g(t) = (k + a(t)^T\delta)t + m + a(t)^T\gamma, \tag{2.31}$$

where k is the growth rate, δ has the rate adjustments, m is the offset parameter, and γ_j is set to -$s_j\delta_j$ to make the function continuous.

Seasonality is provided by Fourier series. Each seasonality is modelled by its own Fourier series. The equation describing one particular seasonality is:

$$y_t = \sum_{n=1}^{N}[a_n cos(\frac{2\pi nt}{P}) + b_n sin(\frac{2\pi nt}{P})], \tag{2.32}$$

where N is the number of Fourier terms used for each particular seasonality.

The events or external regressors, $H(t)$, have to be provided and are also introduced in the model. This component allows to introduce external effects in order to provide more information on the shape of the data. Those elements can be holidays or events in specific dates which can be introduced as a binary variables or also can be extra linear regressors with another time series variable, always taking into account that not only the past values but the future values of those external variables would have to be known. A more detailed information on how exactly all those components are modelled can be found in the paper made by the creators of Prophet [19]. Once the different parts of the model are defined, the different parameters of each component are modeled with some prior distributions and then, prophet fits the model either using L-BFGS [21] to find a maximum a posteriori estimate, or performing a full posterior inference including model parameter uncertainty in the forecast uncertainty.

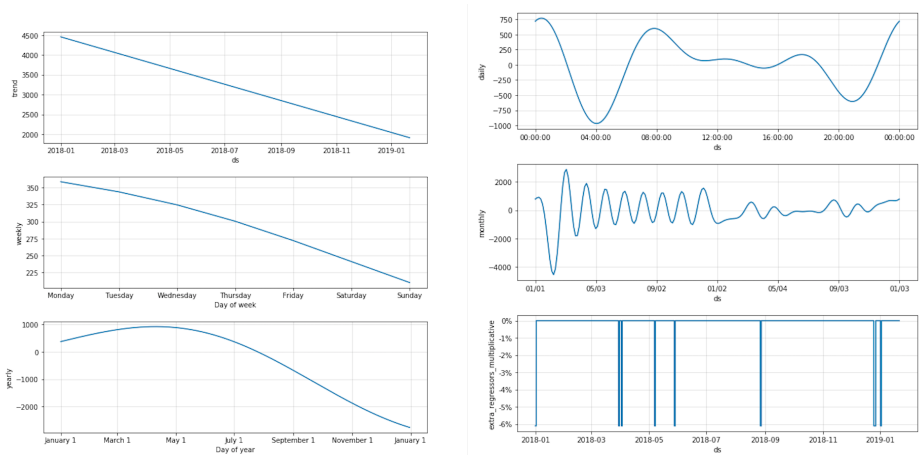


Figure 2.3: Prophet components for a particular location during a one year period of time on hourly data.

Two of the best advantages of Prophet are that being a model made by components allows us to see how each component affects to the forecast separately, and due to its probabilistic formulation, uncertainty intervals can be obtained a part from the forecast of the value.

The particular model build in Prophet for this thesis includes daily, weekly, monthly and yearly seasonalities plus a particular component for each added regressor. This Structural Time Series model turns out to be the one performing the best after trying different combinations of components in Prophet.

2.6.3 STS with Tensorflow Probability

Tensorflow probability [9] provides a new library called *tfp.sts* for forecasting time series using Structural Time Series models as described by Fildes in [11]. The fit of the model is done by Bayesian inference of model parameters using Variational Inference (VI) or Hamiltonian Monte Carlo (HMC), computing both point forecasts and predictive uncertainties.

Therefore, STS models in TFP are built by adding together model components in a similar way as Prophet does. TFP provides a lot of liberty in order to select the different components of the STS. Some of those components are:

- Autoregressive or local linear trend for modelling time series with trend that evolves according to different process.
- Seasonal: Modelling seasonal factors such as daily, weekly, or yearly.

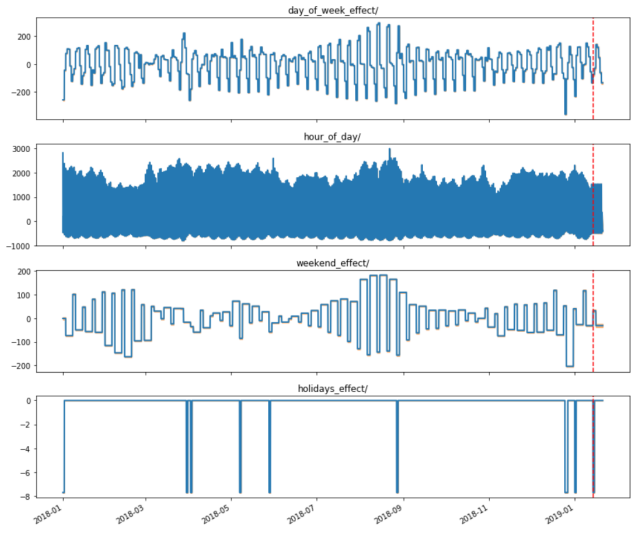


Figure 2.4: STS with TFP components for a particular location during a one year period of time on hourly data.

- Linear regression: To include the relation with external variables in the same period of time. Also can be used to encode specific date effects in the same way Prophet does with holidays.

As it is said in the API docs of Tensorflow [8], a *StructuralTimeSeries* object in TFP represents a declarative specification of a Structural Time Series model, including priors on model parameters. It implements a joint probability model $p(params, y) = p(params)p(y|params)$, where $params$ denotes a list of real-valued parameters specified by the child class, and $p(y|params)$ is a linear Gaussian state space model with structure determined by the child class.

The final STS model build with TFP in this thesis contains a weekly and daily seasonalities, the external regressors and a weekend effect categorical variable modelling weekends. For the daily case, the daily seasonality has been change for a monthly seasonality. This Structural Time Series model turns out to be the one performing the best after trying different combinations of components in TFP.

Chapter 3

Data

For this project two main types of datasets have been used. The first dataset contains the amount of weighted count of people by hour for different locations in London as represented in Table 3.1. These locations are blocks of defined areas mapping all UK but for confidentiality purposes the exact locations and ranges are not provided in the thesis. The weighted count of people is an extrapolation of the count of people taking into account some factors about the location and the users. For simplicity, from now on, we will refer to the weighted count as the regular count of people. The second type of dataset are the external time series variables that were used to fit the multivariate models.

The time series data of the number of people contains the data from three different locations in the city of London and has been provided by the telecommunication company who obtained it aggregating the cell phone data of its users. As a visualization tool, the code developed in this thesis is able to automatically plot the different locations detected into a map as can be seen in Figure 3.1. This is possible thanks to a secondary dataset provided by the company containing a grid of all the UK different locations from which the weighted count can be obtained.

The external data consist of three different datasets: temperature, weather, and holidays. Obtaining these data has not been easy since the company did not have that kind of information. Thus, the temperature and weather have been scraped from the web, concretely from: `https://www.timeanddate.com/weather/`, and the bank holidays dataset has been obtained from `https://www.dmo.gov.uk/media/15008/ukbankholidays.xls`. When using external time series variables we have to take into account that if our aim is to build a forecast for futures dates we will have to use regressors that can be forecast in advance. All variables previously mentioned, temperature, weather and holidays, are variables that can be predicted.

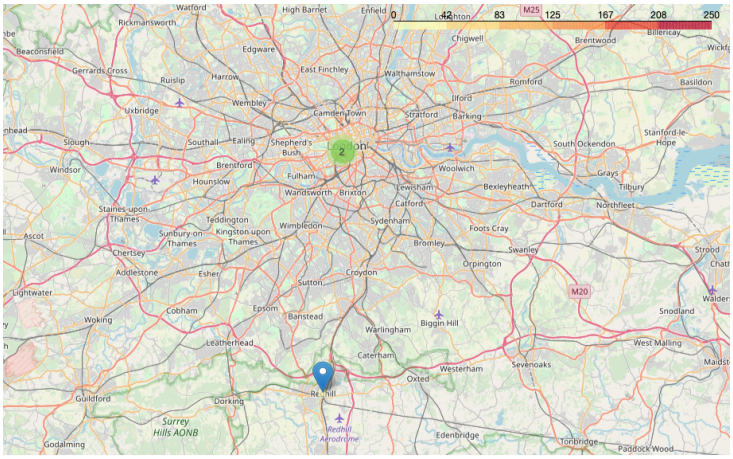


Figure 3.1: The approximate location of the three different locations from which the time series data has been provided. The map and pointers are an output of the the code developed in this thesis using the *folium* and *geopandas* python libraries.

	day_ref	hour	weighted_count	locationId
0	2018-01-01	0	734.670506331837	1001405588125680
10172	2018-01-01	0	557.823580504088	3003405590125670
19859	2018-01-01	0	3783.6519444970300	7007405576125440
24	2018-01-02	0	128.018181741487	1001405588125680
10194	2018-01-02	0	36.386511558398800	3003405590125670
19883	2018-01-02	0	2759.33384349439	7007405576125440
48	2018-01-03	0	195.24205228215100	1001405588125680
10217	2018-01-03	0	28.9895690572537	3003405590125670
19907	2018-01-03	0	2812.6049974395700	7007405576125440
72	2018-01-04	0	196.404908278532	1001405588125680
10241	2018-01-04	0	46.747009669293600	3003405590125670
19931	2018-01-04	0	2718.43382431237	7007405576125440
96	2018-01-05	0	296.692705738197	1001405588125680
10264	2018-01-05	0	52.59494330075070	3003405590125670
19955	2018-01-05	0	2739.4403106232100	7007405576125440
120	2018-01-06	0	454.60398704129100	1001405588125680
10288	2018-01-06	0	54.3618159179428	3003405590125670
19979	2018-01-06	0	2905.8175670882	7007405576125440
144	2018-01-07	0	421.34783185896000	1001405588125680
10311	2018-01-07	0	31.127387919798000	3003405590125670
20003	2018-01-07	0	2959.48139298483	7007405576125440
168	2018-01-08	0	131.594229084924	1001405588125680
10333	2018-01-08	0	45.9852228364725	3003405590125670
20027	2018-01-08	0	2775.9925860699100	7007405576125440

Table 3.1: A sample of the dataset provided with the weighted count of people for three different locations in UK.

Figure 3.2: Walk-forward cross-validation.

To fit the models and evaluate its performance we have split the data into training and testing datasets. Concretely, we have used cross-validation to obtain more accurate results. In time series, the time has to be taken into account to perform cross-validation and for this reason the well-known k-fold cross-validation can not be used. Therefore, we will use a cross-validation variant instead called walk-forward cross-validation [5] which is represented in Figure 3.2. In order to assess which models are better, we have used the Mean Average Error (MAE), the Mean Average Percentage Error (MAPE), and the Mean Average Scaled Error (MASE) as metrics of evaluation.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|, \tag{3.1}$$

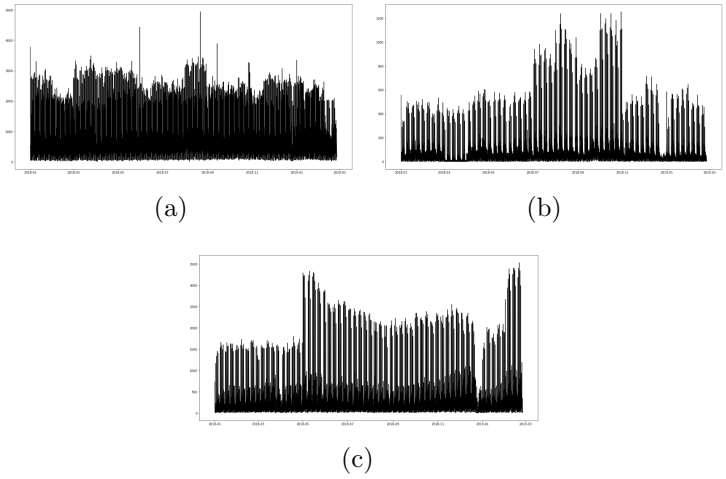
$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right|, \tag{3.2}$$

$$\text{MASE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{\frac{1}{n-m}\sum_{i=m+1}^{n}|y_{i-m} - y_i|}. \tag{3.3}$$

These three are the common metrics of evaluation for time series, for more information about that the reader can take a look at the explanation of Rob J Hyndman in [18] where he claims why MASE can be considered one of the best metrics for time series.

3.1 Time series signal to predict

As stated before, the time series data that we want to predict is the count of people for three different locations in London during a period of time of



(a) (b)

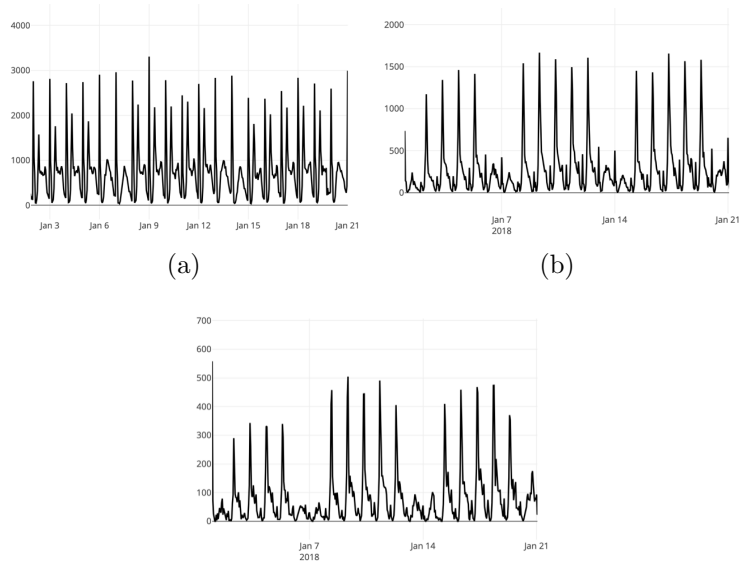
(c)

Figure 3.3: Count of people for different locations: (a) location id: 7007405576125440 , (b) location id: 3003405590125670, (c) location id: 1001405588125680.

14 months starting in January 2018. Each location has a location id, and following the notation of the company, the three locations are named as 7007405576125440, 3003405590125670, and 1001405588125680.

In Figure 3.3, we can see the time series data for each one of those locations during the period of time provided. In Figure 3.4 the same data is provided but with a zoom to better appreciate the weekly and daily seasonality. From the three locations it can be spotted that the 3 signals follow different patterns even though a few similarities can be found. For example, taking a look at Figure 3.5 it can be seen a pattern change during Easter and Christmas for locations 3003405590125670 and 1001405588125680. Moreover, taking a closer look to the zoomed data in Figure 3.6, it can be seen that all three datasets show daily and weekly seasonality.

At some point during the project we decided to work also with an aggregation of the data by day. Even though doing this we loose granularity in time, we also gain in computing timing and the performance of the models changes as we will see in following sections. In Figure 3.7 we can see the representation of the three different locations but with daily periodicity instead of hourly.



(a) (b)

(c)

Figure 3.4: Count of people zoomed for different locations: (a) location id: 7007405576125440 , (b) location id: 3003405590125670, (c) location id: 1001405588125680.

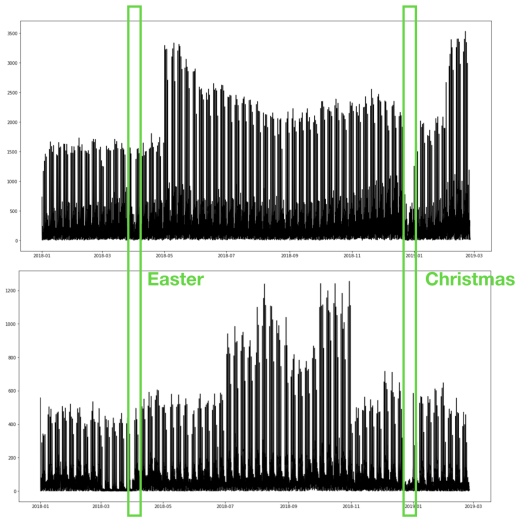


Figure 3.5: Change of patterns due to special events.

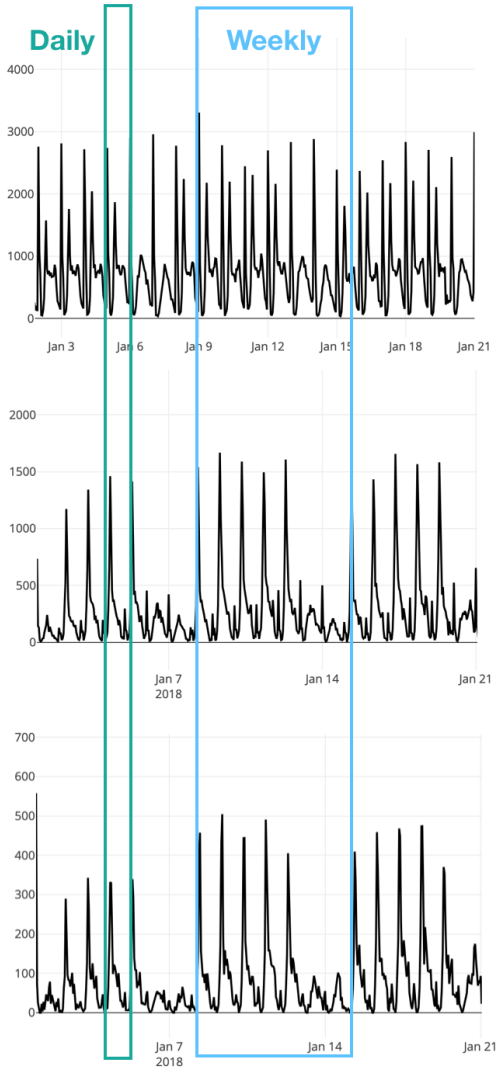
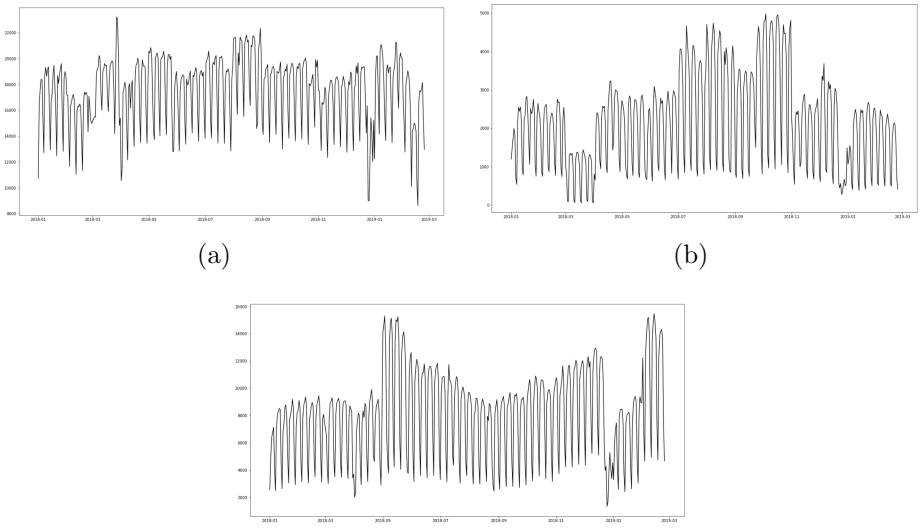


Figure 3.6: Weekly and daily seasonalities.



(a) (b)

(c)

Figure 3.7: Count of people for different locations: (a) location id: 7007405576125440 , (b) location id: 3003405590125670, (c) location id: 1001405588125680.

3.2 Regressors

As it has been stated in previous sections, temperature and weather regressors have been obtained by hour scraping the web and the dates for the holidays regressor have been obtained from a dataset found on internet as well. After cleaning and preparing the data we obtain the three variables in the way we need them. In Figure 3.8, the representation of these three regressors for a period time of one week is plotted. The holidays signal has always the value of zero for this case since none of those represented days are holidays. Temperature is a numerical variable showing the degree Celsius per hour. The null values on temperature have been replaced by the temperature on the day before at the same hour. The weather is a categorical variable with 39 possible classes. Clustering on this variable in order to have less classes under the weather predictor have been tried but without observing any improvements in the performance of the models a part from less computing time.

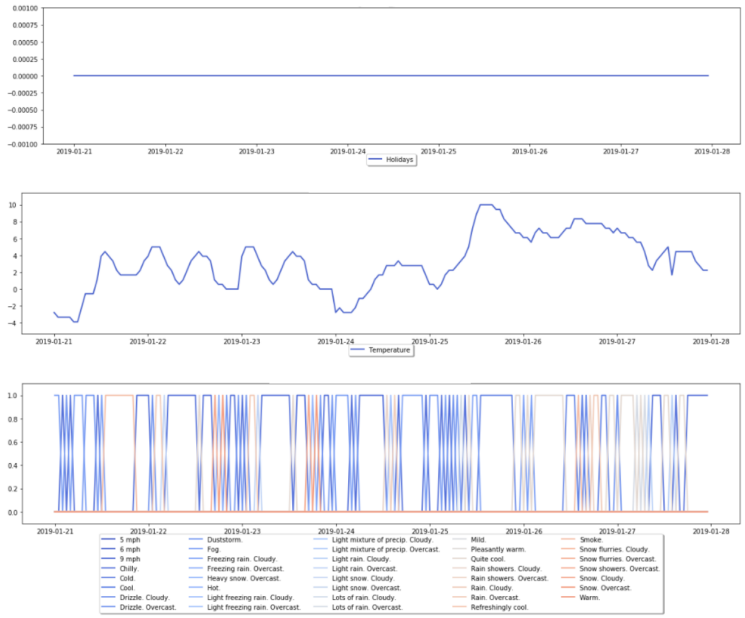


Figure 3.8: Regressors for a particular week.

3.3 Data Transformation

The last attempt with the purpose of improving the models has been to apply some data transformation to the data representing the count of people in each location. Some of the models are known to perform better if the data have a particular distribution or if the data meet some specific conditions. For example, in previous sections when introducing the family of ARIMA models it has been stated why these models perform better with stationary data. Two main transformations to make data stationary are differencing and Box-Cox transformation [12]. With one lag differencing we can eliminate the trend of the data and with a seasonality lag differencing we can eliminate seasonalities. By applying Box-Cox transformation we can overcome heteroscedasticity and convert data to have constant variance. On the other hand, standardization and normalization transformations are highly used in machine learning models since models not only tend to work better but tend to be more interpretable.

Following, in Figures 3.9, 3.10, 3.11, 3.12, and 3.13, we plot the count of people for location 1001405588125680 during the period of time between the 14^{th} of January of 2019 and the 20^{th} of January of 2019 along with its

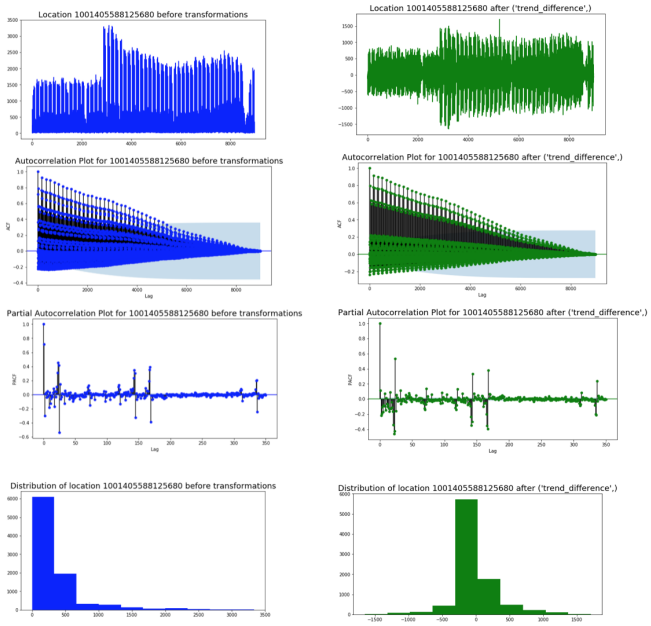


Figure 3.9: Count of people before (blue) and after (green) applying trend difference.

ACF, PACF and data distribution. In blue we have the plots before any transformation and in green after applying them.

For example, taking a look at the ACF and PACF it can be seen how trend (or level) and seasonality are eliminated after trend difference and seasonal difference respectively. In the Box-Cox case, the histogram of the data shows how the transformation changes the distribution to a normal distribution. Regarding normalization and standardization, the shapes of the plots do not change but the values of the signal are scaled correspondingly.

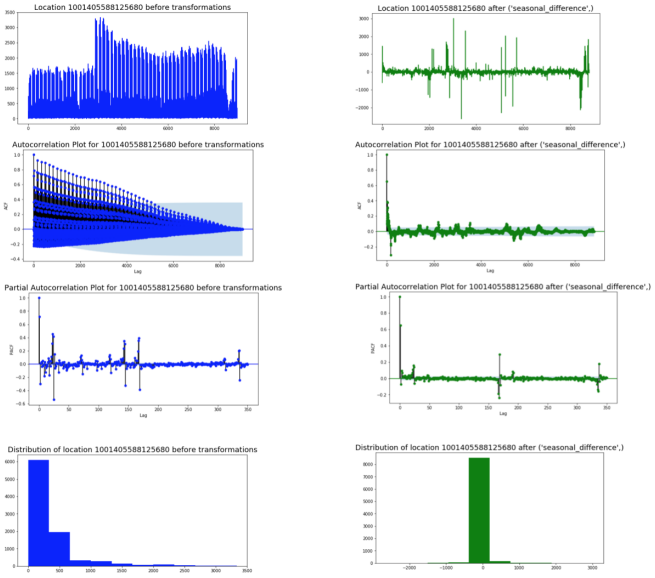


Figure 3.10: Count of people before (blue) and after (green) applying seasonal difference.

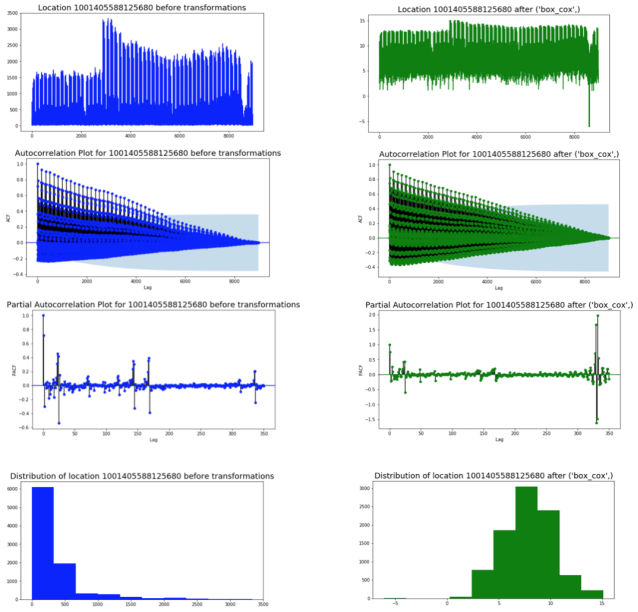


Figure 3.11: Count of people before (blue) and after (green) applying Box-Cox.

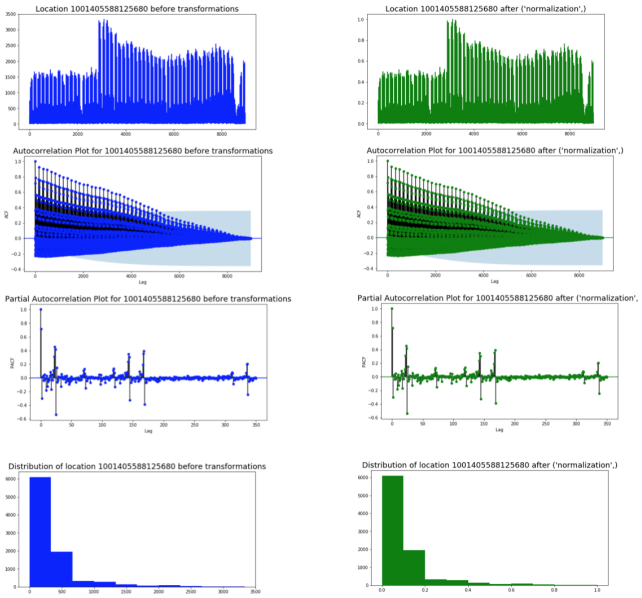


Figure 3.12: Count of people before (blue) and after (green) applying normalization.

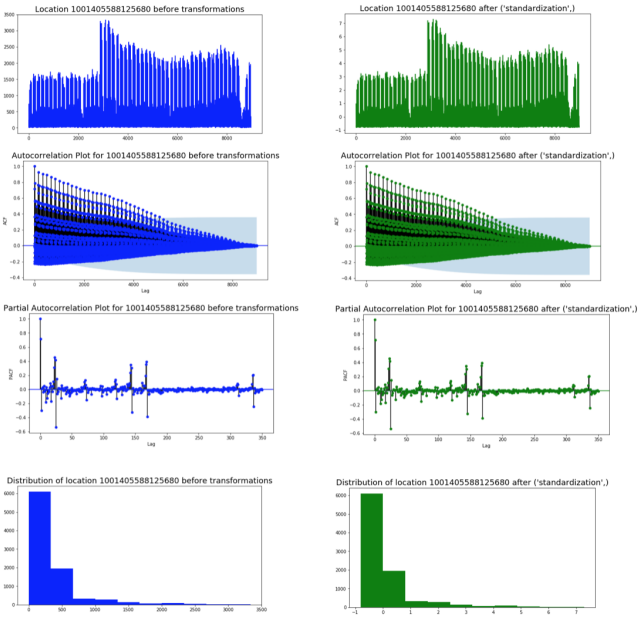


Figure 3.13: Count of people before (blue) and after (green) applying standardization.

Chapter 4

Implementation

In order to identify which models perform better in our data, we have developed an interactive tool to easily evaluate and compare the different models implemented. This tool has been implemented in a Jupyter Notebook coded in Python. The tool provides a user interface with different widgets in order to facilitate the usage for non-technical people. The user interface allows the user to select the locations, models, and several other options to analytically and graphically evaluate the different time series models. The user interface is represented in Figure 4.1. After we select all the options in the user interface and we run the code we could obtain the results to evaluate the models in different ways such as the plot of the forecasts, the analytic results of the different metrics, or some other extra features that will be explained below.

4.1 Input Interface

In this section we describe each of the options of the input interface shown in Figure 4.1.

- **Periodicity**: Daily or hourly depending on the granularity desired on the data.
- **Locations to evaluate**: The code automatically detects all the locations in the dataset selected and gives the option to select one or multiple locations to evaluate.
- **Visualization**: We can select to plot the forecasts of the models with two different python libraries: plotly or matlpotlib.
- **Hours/Days to forecast**: Number of hours or days to forecast depending if we have selected hourly or daily data.

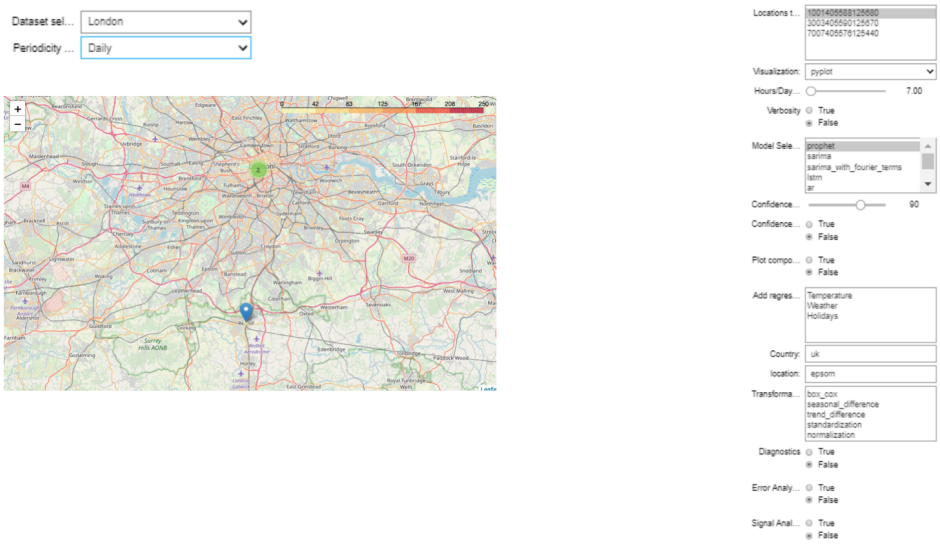


Figure 4.1: User interface to select the locations, models and parameters to evaluate.

- **Verbosity**: Two levels of verbosity.
- **Model Selection**: Select one or more models to evaluate. If we select more than one we will be able to easily compare them.
- **Confidence level (%)**: Some of the models allow not only to forecast a value but an interval. This parameter selects the level of confidence of that interval.
- **Confidence Intervals**: Option to plot or not the forecasts of the models. For those models accepting confidence intervals they will be also plot with a shadow around the forecast.
- **Plot components**: Prophet and STS with TFP give the option to plot the diferent components of the signal and the forecast. If this parameter is selected and Prophet or TFP are used, the components will be plotted as well.
- **Add regressors**: Possibility to add one or more regressors (temperature, weather and holidays). Regressors are going to be used just for the multivariate models.

- **Country**: Name of the country of the location to forecast. Required for the scraping of temperature and weather regressors in case we have selected them in the *Add regressor* option.
- **Location**: Name of the city to forecast. Required for the scraping of the temperature and weather in case we have selected them in the *Add regressor* option.
- **Transformation Selection**: Selection of the transformations to be applied to the data.
- **Diagnostics**: If selected, we will get the plot of the regressors with a vertical red line in those points where the error of the forecast is bigger than the 25%. This utility was included in order to identify if the regressors used were useful or not. The idea is that if the regressors have some kind of strange behaviour when the model has bigger errors it would be worth to try to correct the models with the regressors.
- **Error Analysis**: If selected, the residulas of the forecast will be plot in five different plots :residuals during time line, histogram, scatterplot with predicted values, autocorrelation, and Q-Q Plot.
- **Signal Analysis**: If selected, several plots of the main signal (before and after transformations) are plotted. Specifically, those plots are: signal during time line, histogram, ACF, PACF and stationarity tests.

4.2 Main Outputs

In order to evaluate the models we need to analyze the main outputs of the code implemented. These main outputs are basically two: the plot of the forecasts of the models and the graphical and analytic evolution of the metrics chosen. In Figure 4.2 the plots for the forecasts of all the models for a particular week on location 7007405576125440 are represented. In Table 4.1, the averages of the different cross-validation attempts for all the metrics and for the different locations and models are represented. Finally, in Figure 4.3, the evolution of the three metrics during six following cross-validation attempts on location 00140558812568 can be seen.

4.3 Additional Outputs

A part from the main outputs, some additional outputs can be plotted if selected in the user interface. These additional outputs have been created in

Locations

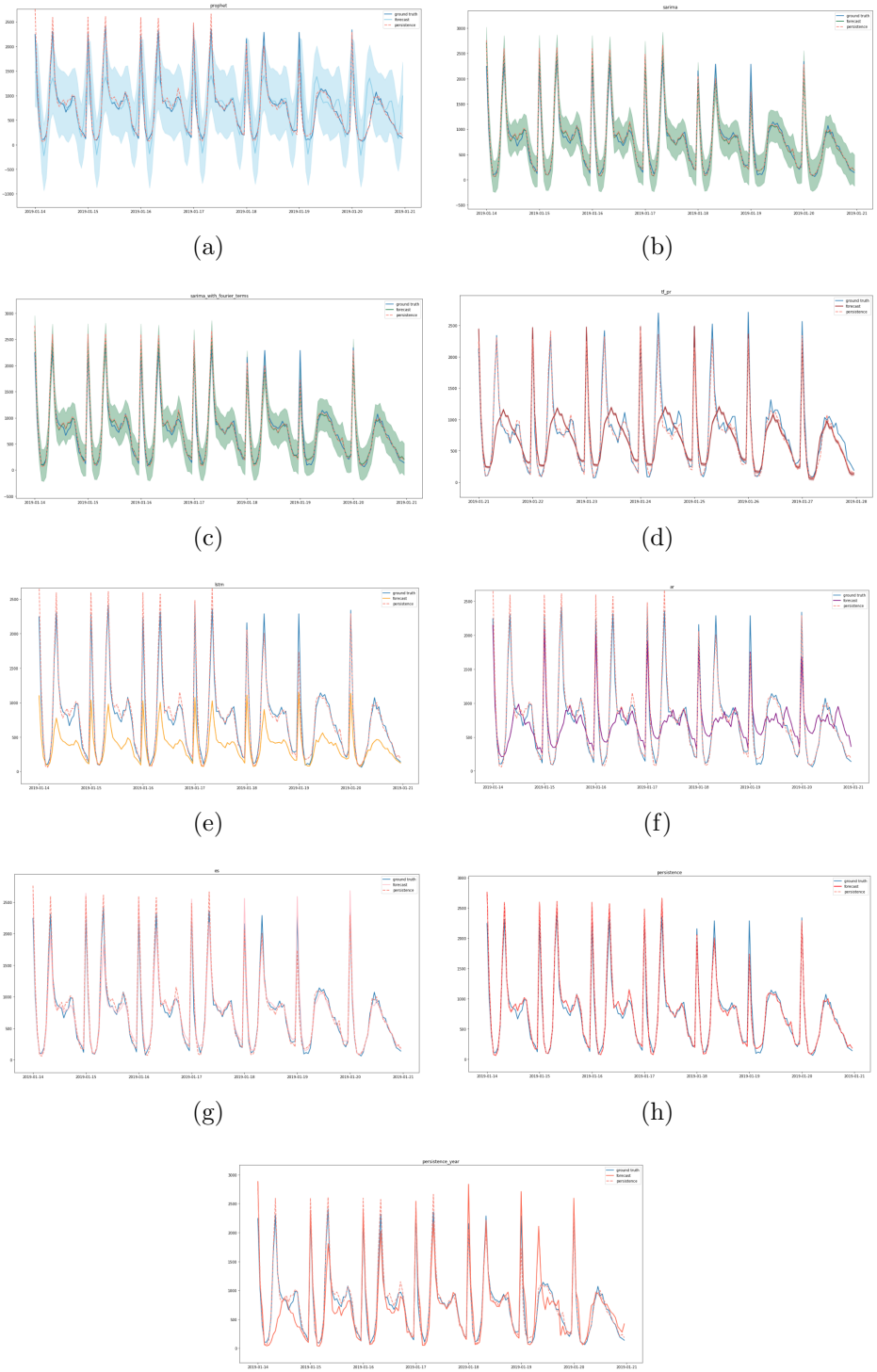
Models

Metrics

	avg_MAPE for 1001405588125680	avg_MAPE for 3003405590125670	avg_MAPE for 7007405576125440	Avg_MAPE_all_locations
persistence_year	0.388214	0.744297	0.151245	0.427919
lstm	0.241400	0.372994	0.118183	0.244192
prophet	0.243998	0.165944	0.103191	0.171044
ar	0.190703	0.202612	0.115159	0.169492
sarima_with_fourier_terms	0.186460	0.156636	0.142876	0.155324
tf_pr	0.154549	0.151400	0.124884	0.143611
persistence	0.133206	0.128012	0.145061	0.135426
es	0.145584	0.137390	0.107920	0.131298
sarima	0.139716	0.125994	0.107863	0.124524
	avg_MASE for 1001405588125680	avg_MASE for 3003405590125670	avg_MASE for 7007405576125440	Avg_MASE_all_locations
persistence_year	5.013031	5.332984	1.198741	3.848252
lstm	2.268529	4.712264	0.947220	2.642671
prophet	2.979770	1.417762	0.616197	1.671243
es	1.962028	1.447276	0.723928	1.377744
ar	1.727551	1.389678	0.772905	1.296711
sarima	1.669315	1.293773	0.764342	1.242477
tf_pr	1.611109	1.263710	0.849588	1.241468
sarima_with_fourier_terms	1.215777	1.284149	0.992748	1.164225
persistence	1.000000	1.000000	1.000000	1.000000
	avg_MAE for 1001405588125680	avg_MAE for 3003405590125670	avg_MAE for 7007405576125440	Avg_MAE_all_locations
persistence_year	3132.572326	698.245434	2308.140617	2046.319459
lstm	1829.409406	666.004932	1879.288560	1458.234299
sarima_with_fourier_terms	1403.787740	243.820316	2255.577118	1301.061725
prophet	2084.640545	210.992020	1477.013483	1257.548676
persistence	1201.400477	209.040183	2261.490625	1223.977095
tf_pr	1364.730491	220.944588	1935.173851	1173.616310
ar	1485.010295	263.037727	1746.900285	1164.982769
es	1378.001734	221.761759	1847.054058	1082.272517
sarima	1331.170226	212.537650	1822.791302	1055.499726

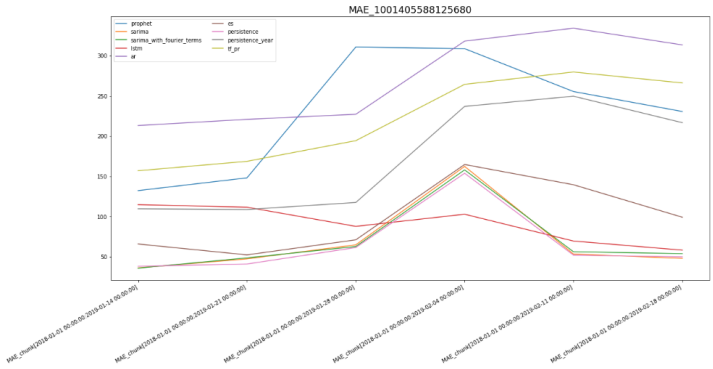
Table 4.1: Average metrics for all the models for all three locations.

order to provide tools to better understand the models and to facilitate its improvement. Those additional outputs are: plot components, signal analysis, error analysis, diagnostics, and computing time. An example of each one of them can be seen in Figures 4.4, 4.5, 4.6, 4.7, and 4.8 respectively. Plot components represents each component of the STS model used by Prophet or TFP. With signal analysis we can basically take a look at the distribution of the signal and check if it is stationary. Error analysis is used to find if residuals follow normal distributions, and diagnostics serve to spot correlation between regressors and errors on the models. Finally, the analysis of computing time can give us an idea of the differences in terms of time needed to run each model.

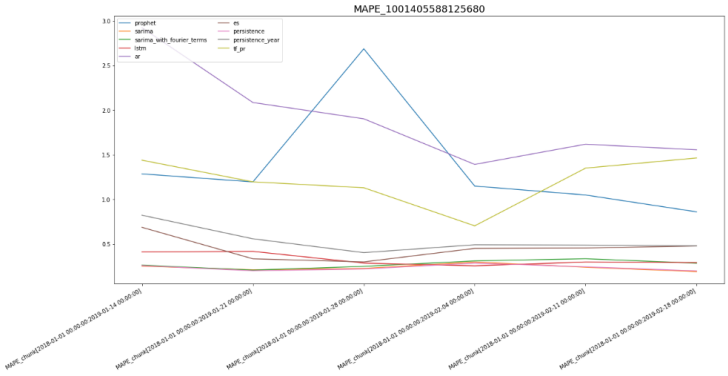


(a) (b) (c) (d) (e) (f) (g) (h) (i)

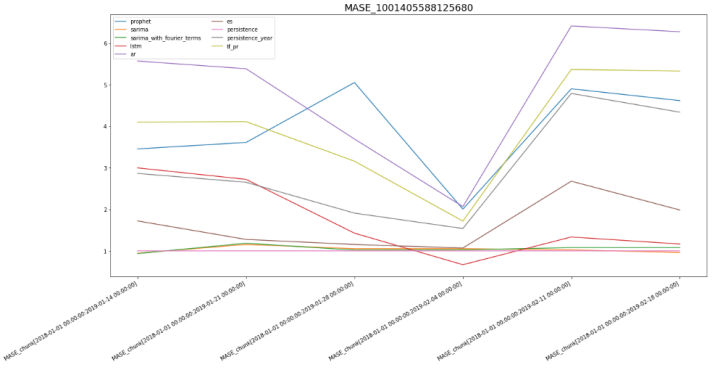
Figure 4.2: Forecast plot for a particular cross-validation attempt on location 7007405576125440 for each of the models implemented: (a) Prophet , (b) SARIMA, (c) SARIMA with fourier terms, (d) LSTM, (e) AR, (f) ES, (g) Persistence weekly, (h) Persistence yearly.



(a)



(b)



(c)

Figure 4.3: Evolution of the three metrics during six following cross-validation attempts for location 1001405588125680: (a) MAE , (b) MAPE, (c) MASE.

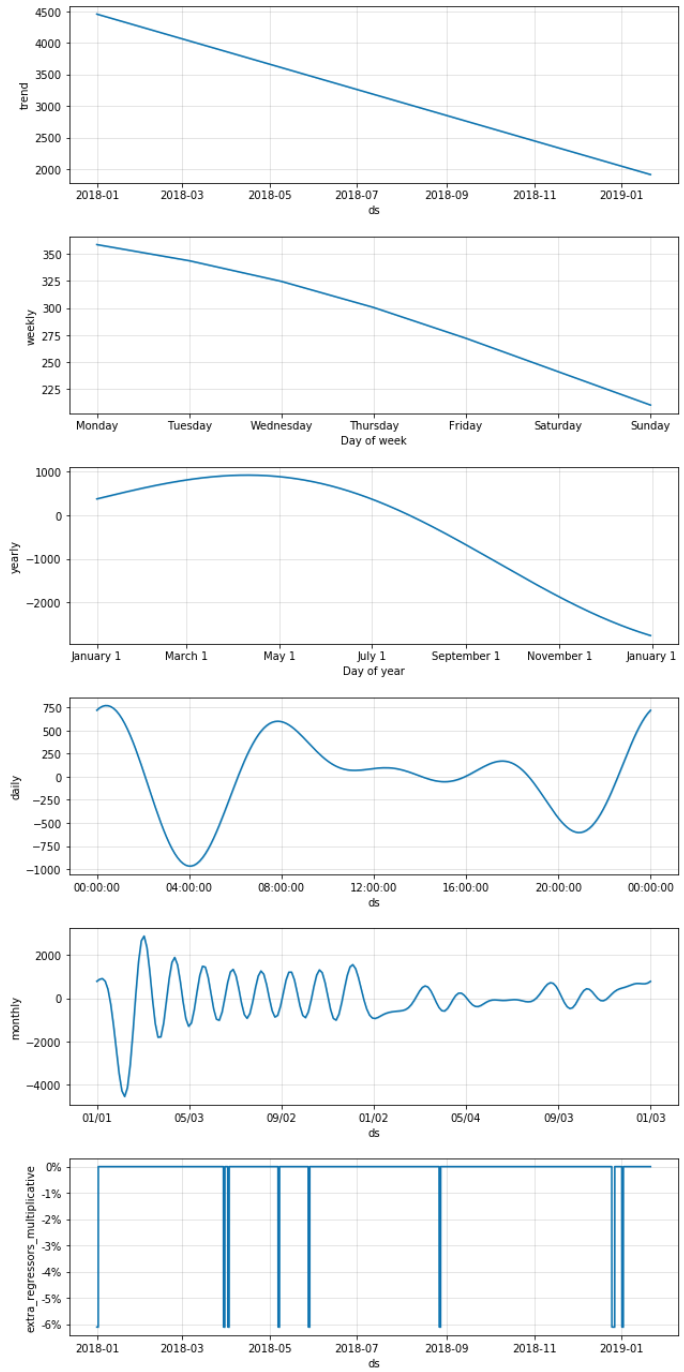


Figure 4.4: Decomposition of the different components of the signal forecast with Prophet.

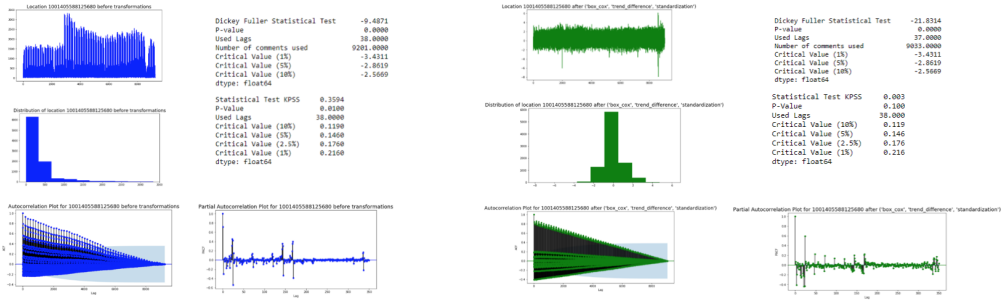


Figure 4.5: Signal Analysis, before and after transformations, for the training data on location 1001405588125680 for a particular cross-validation attempt. The specific transformations performed are: Box-Cox, trend difference and standardization.

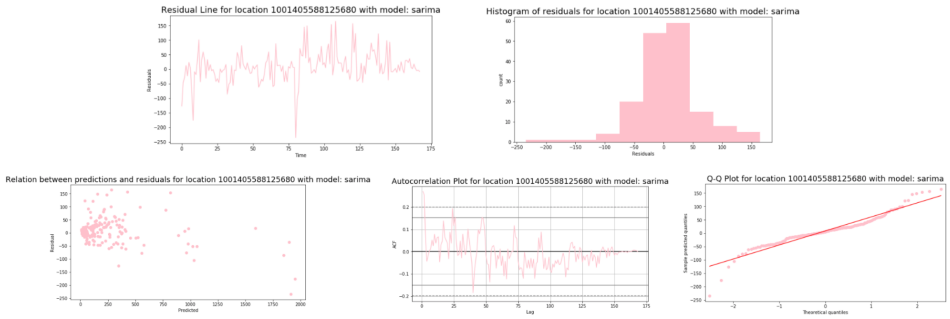


Figure 4.6: Error Analysis for a particular cross-validation attempt on location 1001405588125680 with sarima model.

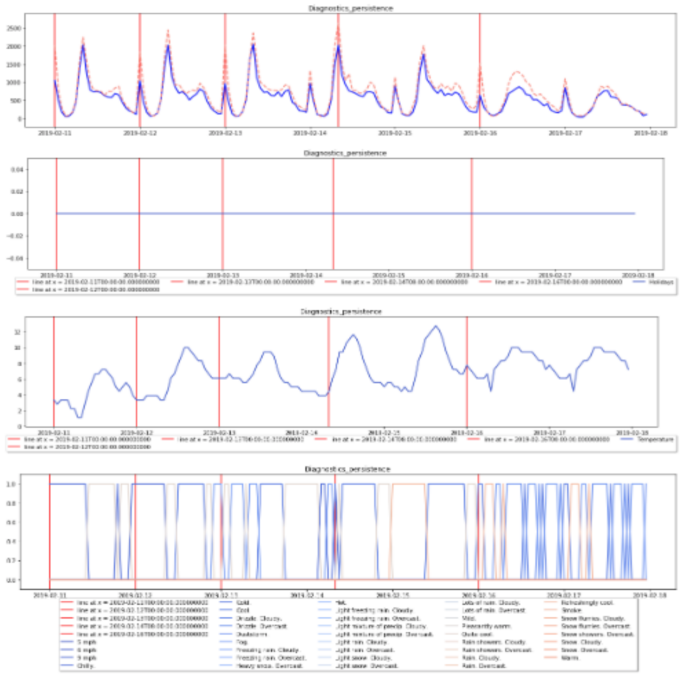


Figure 4.7: Diagnostic for persistence model for a particular cross-validation attempt.

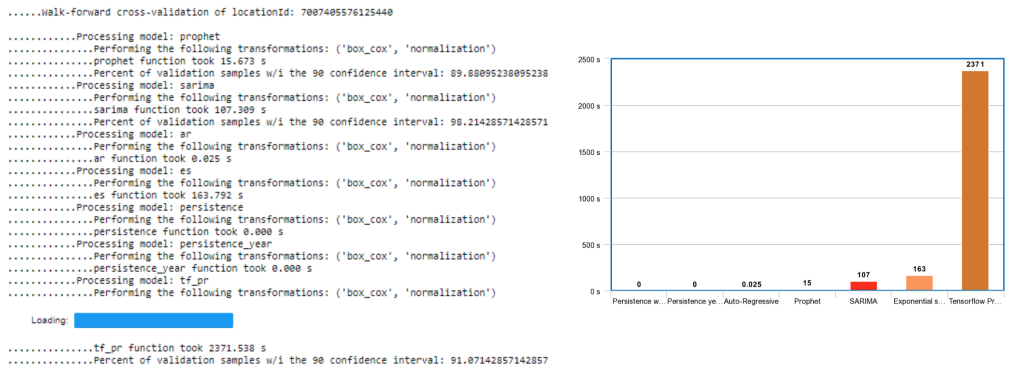


Figure 4.8: Computing timing of all the models for a particular cross-validation attempt on location 7007405576125440.

Chapter 5

Results

In this part of the thesis, the results of applying the models to the data provided are shown. Since there are no direct benchmarks, we defined a baseline model in order to have something to compare with. The baseline model used is the Persistence model explained in previous sections. As a reminder, the Persistence algorithm basically persists the observations for the same time in the previous season. In other words, our prediction for time t, will be $\hat{y}_t = y_{t-k}$ being k the seasonality.

This project has basically focused on building a tool to easily test and compare the different models with time series data rather than to focus in tuning one particular model for a particular time series signal. All the best models for time series that we discovered in our research have been implemented and posteriori evaluated on the three different signals provided in the data of this thesis. It is important to emphasize that each of these models have a number of different parameters to be tuned and although a superficial tuning has been performed, the proper exhaustive tuning of all the models for each of the signals has been considered out of the scope of the thesis.

5.1 Evaluation

As mentioned before, to analytically compare the performance of the models we have used MAE, MAPE and MASE as metrics of evaluation for our models. Moreover, in order to obtain more accurate results for these metrics, all the results have been obtained with a walk-forward cross-validation.

The dataset that we want to evaluate contains fourteen month of data representing the amount of people by hour in three different locations in London. In particular, to compare the models, we have focused in the capacity of the models to predict the following week. That means that in the case

of the hourly data we have forecast the next 168 hours, and for the case of daily periodicity the following seven days. For doing this, we have performed six walk-forward cross-validation attempts taking as a test set each one of the last six weeks of the dataset, starting with the week from 14^{th} to 21^{st} of January 2019 and finishing on the week from 18^{th} to 25^{th} of February 2019.

First, we test all the models for each one of the three locations with the hourly data and then we perform the same experiments with the daily data. After this, for both cases, we select the location were we have obtained better metrics and we take a closer look at it, meaning that we perform a grid search of the combination of model, regressors and data transformation leading to better metrics. Specifically, we have focused on improving the metric MASE. The procedure followed for both cases have been to start evaluating the models without data transformations and without regressors. Then, applying the different transformations and combinations of those transformations leading to better MASE. And finally, applying the regressors to the multivariate models and in case they help the models in terms of MASE, we combine them with the best data transformations hoping to get better performance.

In this thesis we have also measured computing timing of the models even though our priority was in performance. The code implemented is able to provide computing timing for every model and even though we are not going into detail on that, it is worth to mention the bullet points about what we saw. Thus, the model taking more time without regressors is the Structural Time Series model implemented with TFP followed by SARIMA models, but as we start to include regressors, it turns out that SARIMA models start to perform really slow in comparison.

5.2 Results

5.2.1 Hourly data

Starting for the hourly data, the results of applying all the models to the three locations are shown in Figures 5.1, 5.2, 5.3 and Table 5.1. Figures 5.1, 5.2 and 5.3 represent the evolution of the three metrics for all the cross-validation attempts on all locations for all the models. Table 5.1 shows the averages of the values plotted in the other three figures. From these plots, we can devise that in the hourly case the models implemented do not seem to perform much better than the Persistence (weekly) model. From Figures 5.1, 5.2 and 5.3 we can see that the performance of the models really depends on the particular cross-validation attempt being forecast. Moreover, it can be

	avg_MAPE for 1001405588125680	avg_MAPE for 3003405590125670	avg_MAPE for 7007405576125440	Avg_MAPE_all_locations
prophet	3.337866	4.328961	1.188780	2.951869
ar	1.913668	3.338032	1.036623	2.096108
tf_pr	1.214218	2.267747	0.569139	1.350368
persistence_year	0.539972	1.299084	0.338816	0.725957
es	0.450099	0.595151	0.296164	0.447138
sarima_with_fourier_terms	0.274391	0.498512	0.233816	0.335573
persistence	0.233837	0.466807	0.208215	0.302953
sarima	0.235651	0.445827	0.203919	0.295132
lstm	NaN	NaN	NaN	NaN

	avg_MASE for 1001405588125680	avg_MASE for 3003405590125670	avg_MASE for 7007405576125440	Avg_MASE_all_locations
prophet	7.517283	5.227627	3.627057	5.457322
ar	4.899411	4.539943	2.690290	4.043215
tf_pr	3.991634	3.406659	2.140963	3.179752
persistence_year	3.015919	2.501691	1.684799	2.400803
es	1.648283	1.240970	1.139967	1.343073
sarima_with_fourier_terms	1.055327	1.002725	0.994460	1.017504
sarima	1.033169	0.986156	0.990911	1.003412
persistence	1.000000	1.000000	1.000000	1.000000
lstm	NaN	NaN	NaN	NaN

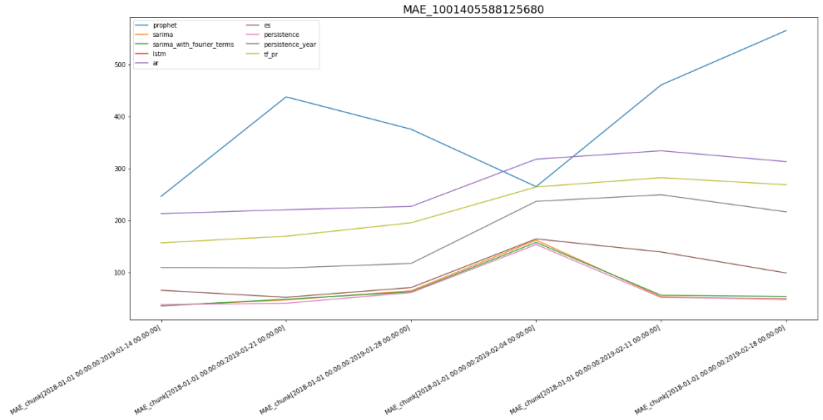
	avg_MAE for 1001405588125680	avg_MAE for 3003405590125670	avg_MAE for 7007405576125440	Avg_MAE_all_locations
prophet	391.937073	80.380415	379.640014	283.985834
ar	271.142511	68.550951	289.516358	209.736607
tf_pr	223.128698	51.504264	233.976545	169.536503
persistence_year	173.222626	37.778592	181.735835	130.912351
es	98.828669	18.654786	128.049759	81.844405
sarima_with_fourier_terms	69.276143	15.226596	113.209388	65.904042
sarima	68.732921	14.996996	112.737973	65.489297
persistence	66.129770	15.254286	113.938336	65.107464
lstm	NaN	NaN	NaN	NaN

Table 5.1: Average of all cross-validation attempts for hourly data on all the locations.

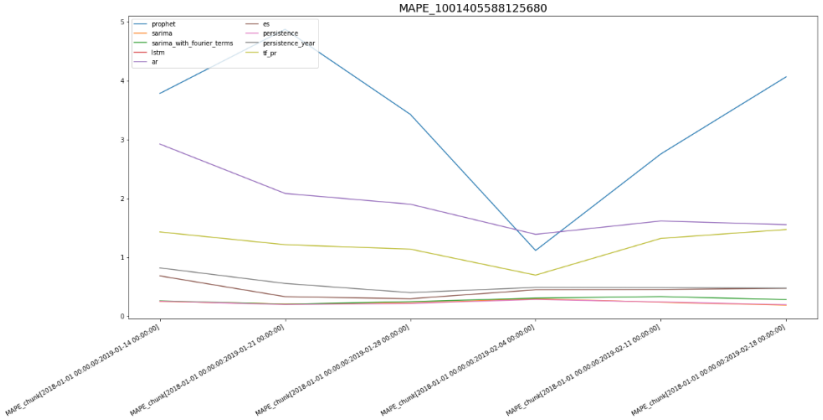
seen that even though some models tend to follow similar curves, meaning that they perform better for the same cross-validation attempts, others really do not show much correlation. It is also important to state that for the hourly case we have omitted the LSTM case since for some cross-validation attempts the code crashed. This should be analyzed in more detail as it is stated in future sections but has been considered out of the scope of this project due to the lack of time.

For simplicity we will focus on analyzing one of the metrics, the MASE. MASE gives values bigger than 1.0 for those models performing worse than the Persistence model and less than 1.0 for those performing better. In our case, taking a look at Table 5.1, we can see that just for location 7007405576125440 and 3003405590125670 SARIMA models are able to slightly outperform the Persistence model even though the difference is minuscule.

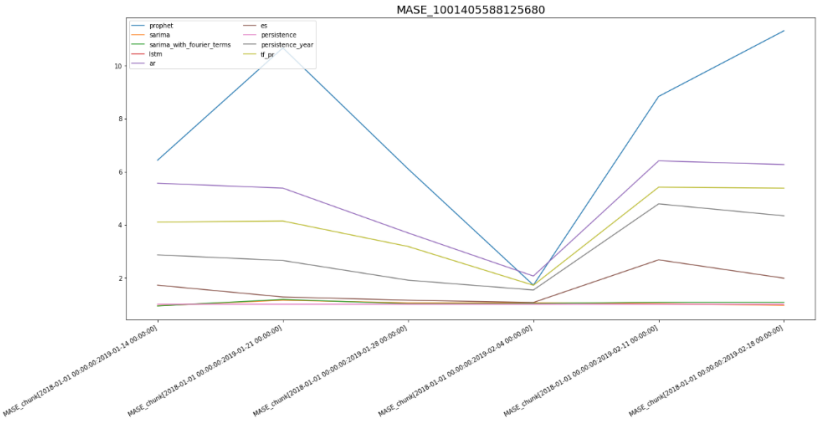
At first look at table 5.1, results do not seem very promising. Now, we are going to focus more in deep in location 7007405576125440 which seems to be the one where our models perform better in general. Specifically, we will



(a)

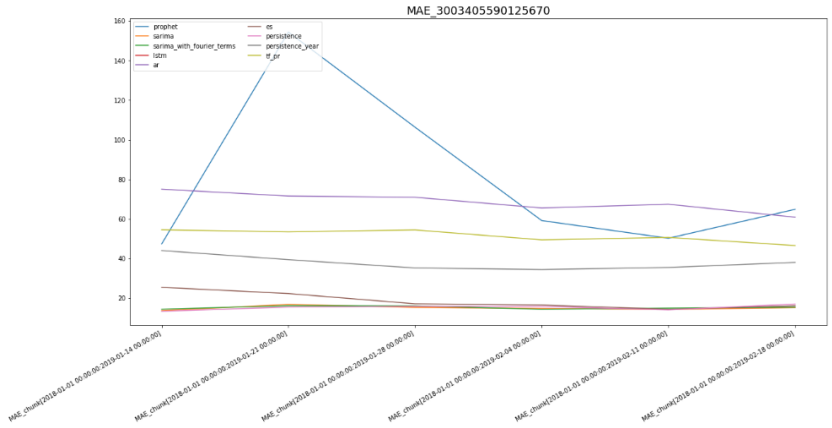


(b)

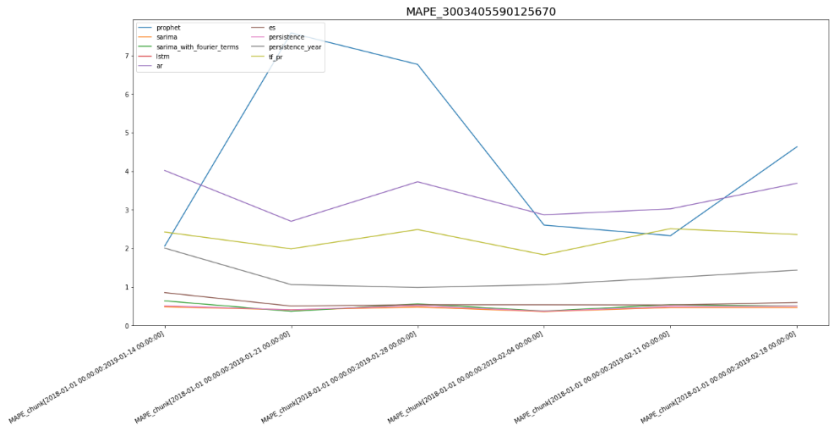


(c)

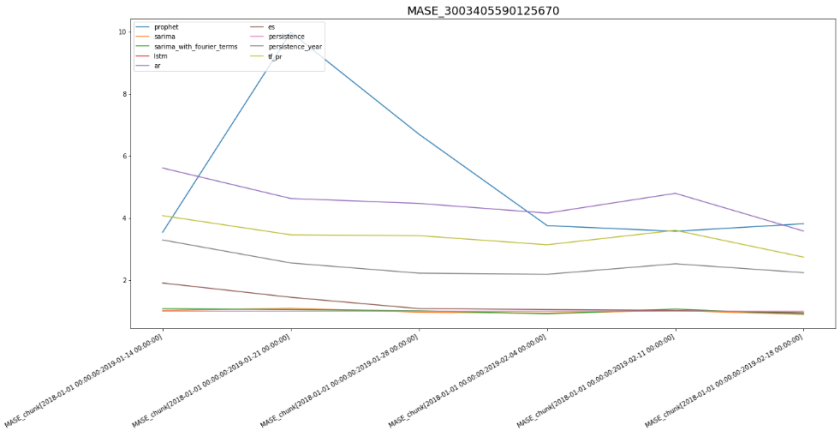
Figure 5.1: Evolution of the three metrics during six following cross-validation attempts for hourly data on location 1001405588125680: (a) MAE, (b) MAPE, (c) MASE.



(a)

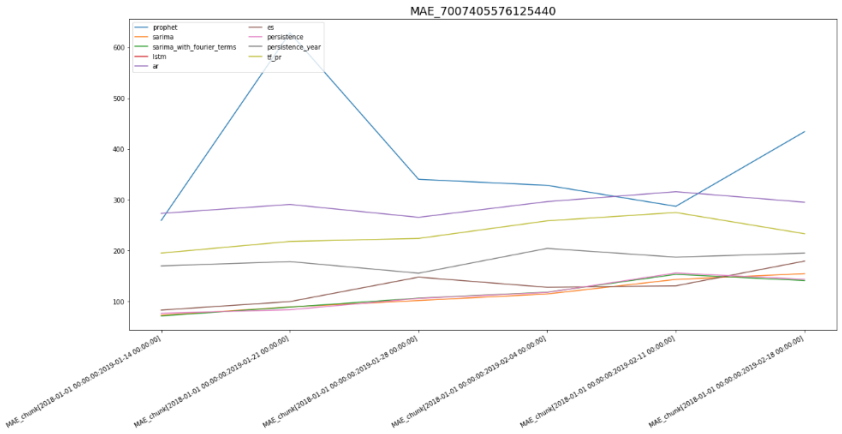


(b)

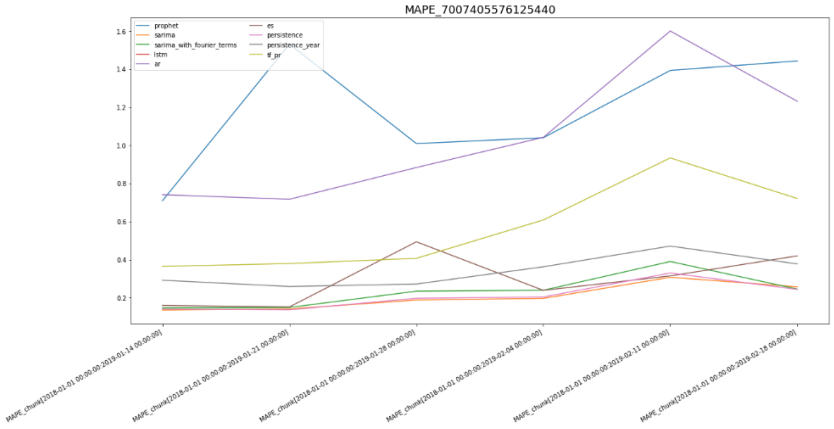


(c)

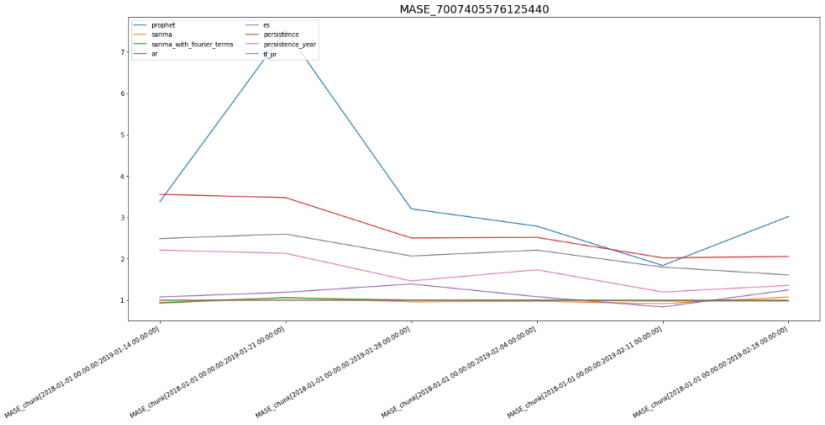
Figure 5.2: Evolution of the three metrics during six following cross-validation attempts for hourly data on location 3003405590125670: (a) MAE, (b) MAPE, (c) MASE.



(a)



(b)



(c)

Figure 5.3: Evolution of the three metrics during six following cross-validation attempts for hourly data on location 7007405576125440: (a) MAE, (b) MAPE, (c) MASE.

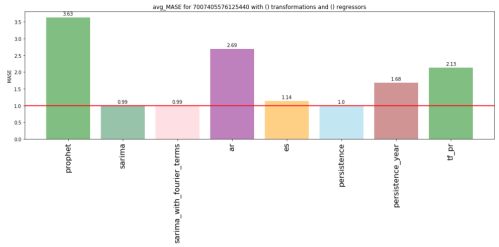


Figure 5.4: Average MASE of all cross-validation attempts for each model on location 7007405576125440 for the hourly data.

analyze more in detail this location applying different data transformations and using regressors on those models which permit it. Results of metric MASE for all the models without transforamtions and without regressors are plotted in Figure 5.4.

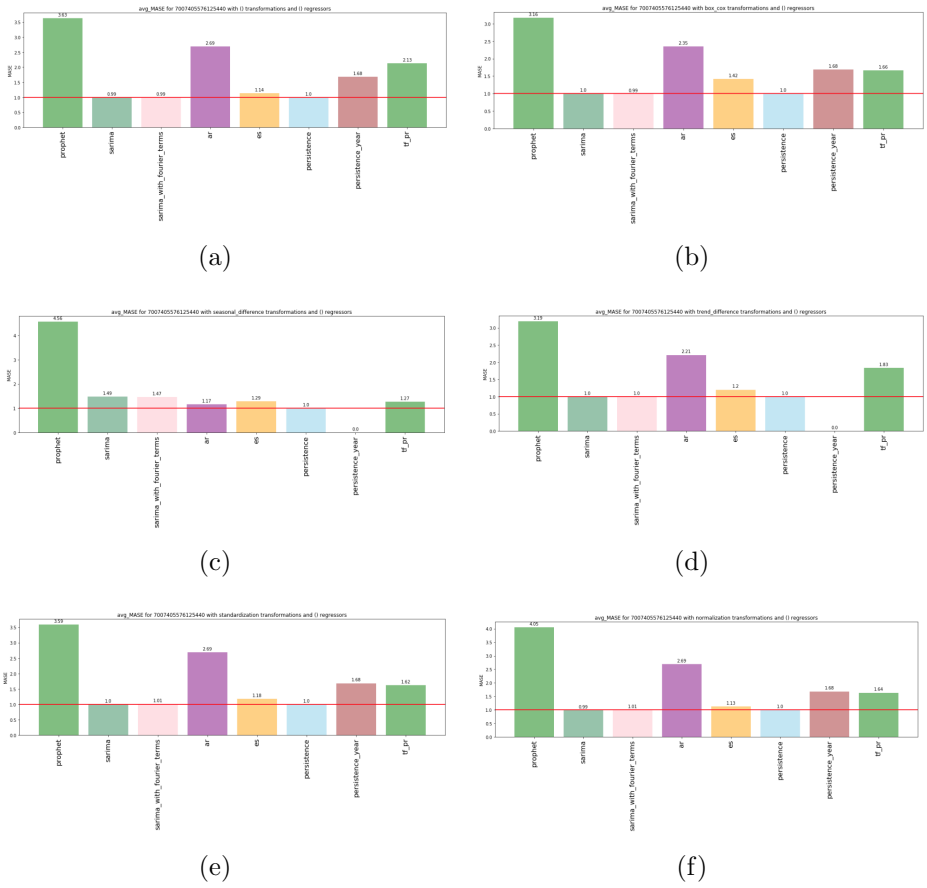
The results of the models after applying the different transformations one at a time can be seen at Figure 5.5. Some transformations improve the MASE of some models but no transformations seem to improve SARIMA models which are the ones performing better.

Since transformations do not improve the models we will take a look to the regressors. For the hourly data we have three external regressors as we explained in previous sections: holidays, temperature, and weather. Results after including the regressors to the models who permit it are represented in Figure 5.6. Taking into account that the models accepting regressors are: Prophet, SARIMA, SARIMA with Fourier Terms, and TensorFlow Probability, if we compare them with the results whithout using regressors in Figure 5.4, not big differences can be spotted. Particularly, for our best models which are SARIMA models, regressors do not seem to make any difference.

5.2.2 Daily data

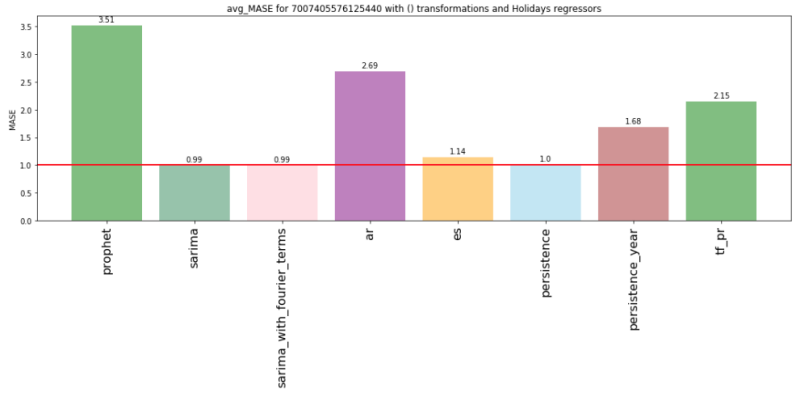
Results of the evolution through each cross-validation attempt of the three metrics after applying all the models to each of the three locations for the daily data are shown in Figures 5.7, 5.8, and 5.9. In Table 5.2 the averages of each metric for each location and model are presented as well as the averages for all the locations evaluated.

As it happened for the hourly data, the performance on the evolution of the metrics really depends on the particular cross-validation attempt. At first look at Table 5.2, results do not seem very promising. As before, we are going to concentrate on analyzing the MASE metric and rapidly can be

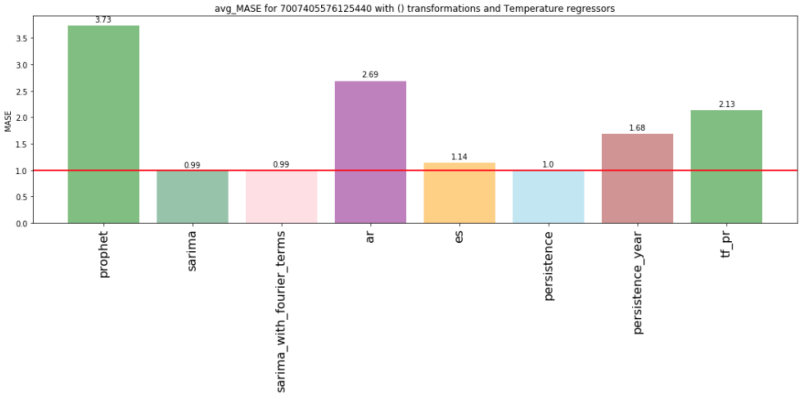


(a) (b) (c) (d) (e) (f)

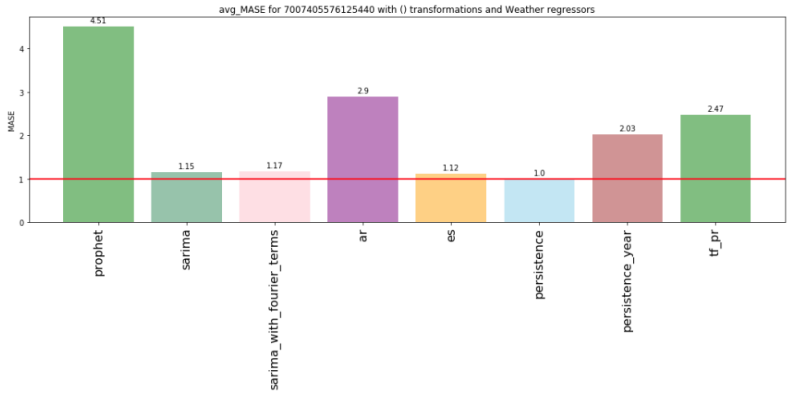
Figure 5.5: MASE metric for all the models and different data transformation for hourly data on location 7007405576125440: (a) Without data transformations , (b) Box-Cox, (c) Seasonal difference, (d) Trend difference, (e) Standardization, (f) Normalization.



(a)

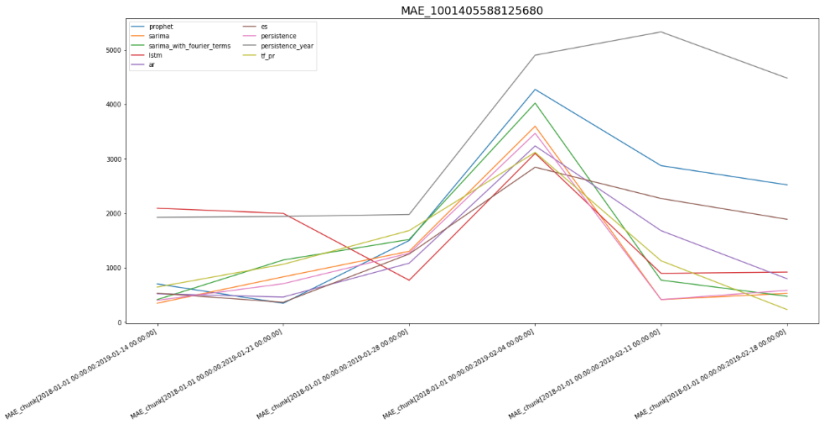


(b)

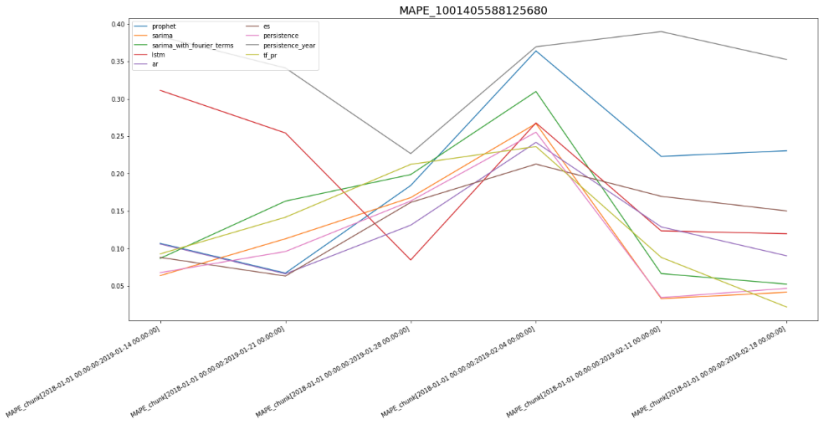


(c)

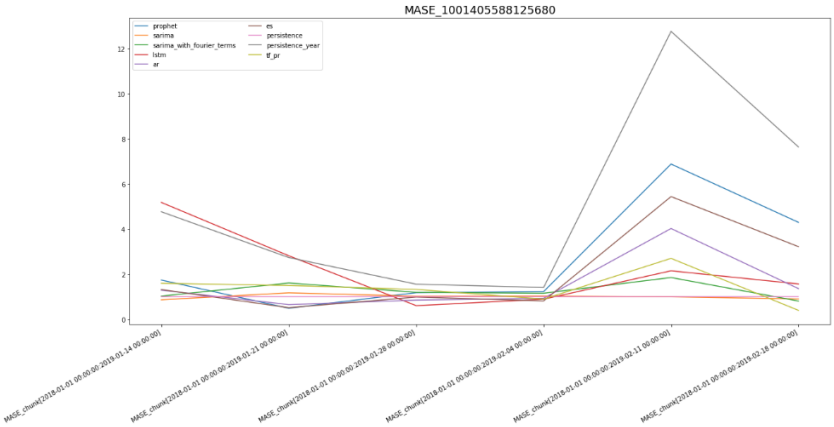
Figure 5.6: MASE metric for all the models without data transformation but using the holiday regressor on the multivariate models for hourly data on location 7007405576125440: (a) holidays, (b) temperature, (c) weather.



(a)

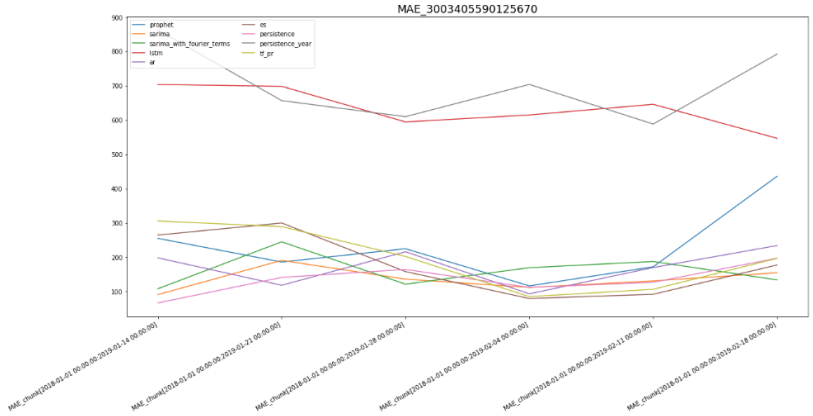


(b)

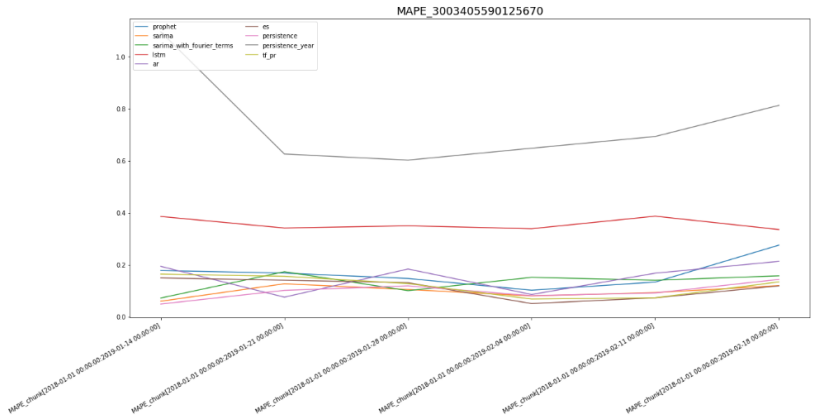


(c)

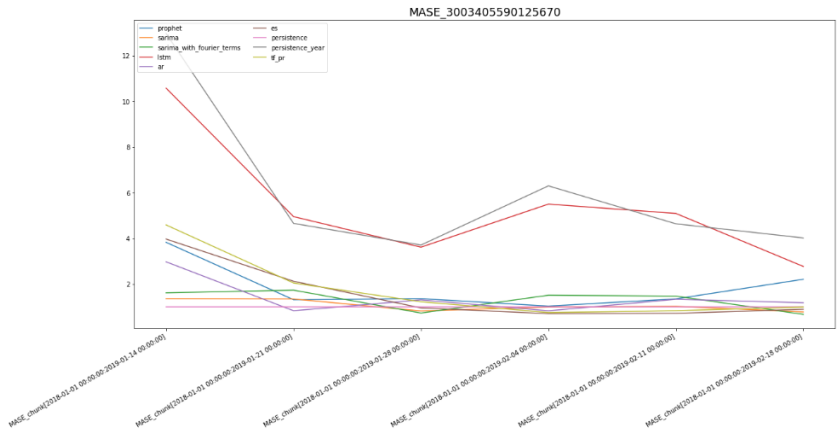
Figure 5.7: Evolution of the three metrics during six following cross-validation attempts for daily data on location 1001405588125680: (a) MAE, (b) MAPE, (c) MASE.



(a)

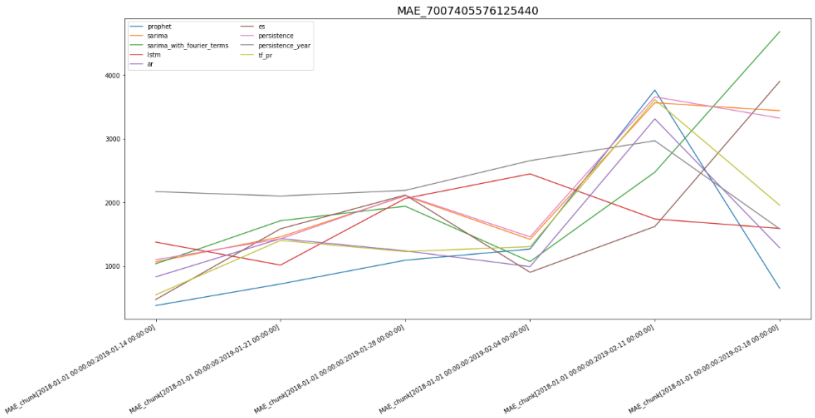


(b)

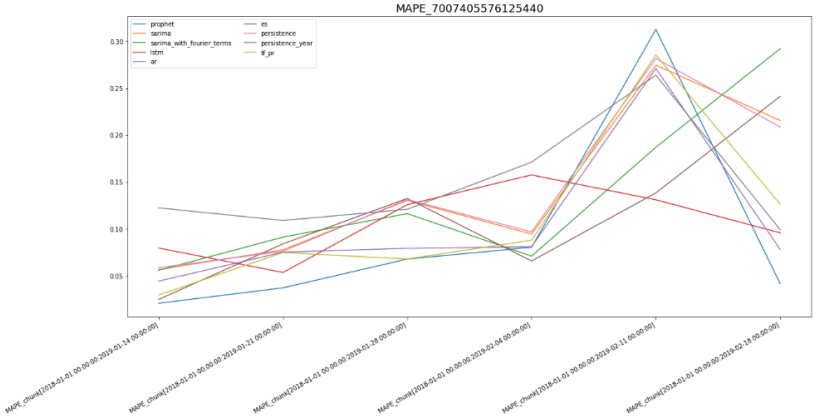


(c)

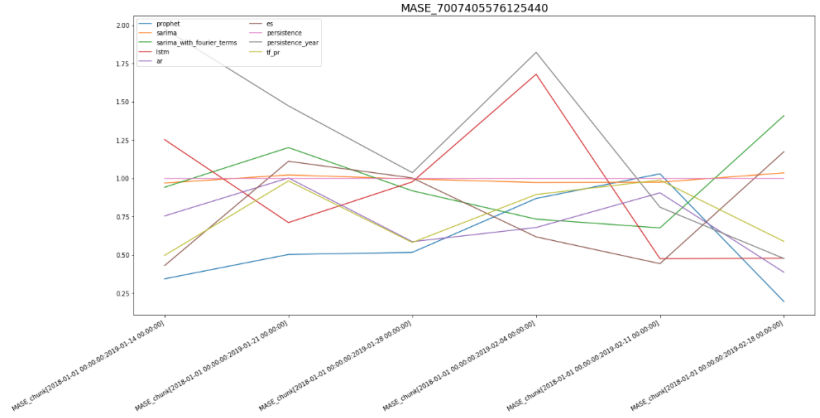
Figure 5.8: Evolution of the three metrics during six following cross-validation attempts for daily data on location 3003405590125670: (a) MAE, (b) MAPE, (c) MASE.



(a)



(b)



(c)

Figure 5.9: Evolution of the three metrics during six following cross-validation attempts for daily data on location 7007405576125440: (a) MAE, (b) MAPE, (c) MASE.

	avg_MAPE for 1001405588125680	avg_MAPE for 3003405590125670	avg_MAPE for 7007405576125440	Avg_MAPE_all_locations
persistence_year	0.343867	0.746314	0.147858	0.412680
lstm	0.193539	0.356403	0.107320	0.219088
prophet	0.195951	0.167567	0.093476	0.152332
sarima_with_fourier_terms	0.146247	0.132822	0.135677	0.138249
ar	0.127345	0.153415	0.104919	0.128560
es	0.140895	0.110803	0.114568	0.122089
tf_pr	0.132172	0.120699	0.112078	0.121650
sarima	0.114338	0.097705	0.141690	0.117911
persistence	0.110570	0.097880	0.142291	0.116913

	avg_MASE for 1001405588125680	avg_MASE for 3003405590125670	avg_MASE for 7007405576125440	Avg_MASE_all_locations
persistence_year	5.149093	6.050939	1.265847	4.155293
lstm	2.205511	5.421509	0.928750	2.851924
prophet	2.640100	1.854723	0.575899	1.690241
es	2.052980	1.566152	0.796171	1.471768
tf_pr	1.406007	1.745053	0.755149	1.302070
ar	1.522257	1.414912	0.718869	1.218680
sarima_with_fourier_terms	1.279451	1.294925	0.979742	1.184706
sarima	1.002925	1.060420	0.994901	1.019415
persistence	1.000000	1.000000	1.000000	1.000000

	avg_MAE for 1001405588125680	avg_MAE for 3003405590125670	avg_MAE for 7007405576125440	Avg_MAE_all_locations
persistence_year	3427.721049	702.533958	2277.358938	2135.871315
lstm	1631.432830	634.239058	1704.096096	1323.255995
sarima_with_fourier_terms	1393.106361	160.771844	2151.578651	1235.152285
prophet	2038.055661	231.651984	1310.848830	1193.518825
sarima	1172.813248	135.793790	2174.729178	1161.112072
es	1527.712464	178.475303	1765.369839	1157.185869
persistence	1142.300743	134.555357	2179.265871	1152.040657
tf_pr	1313.193159	197.469321	1674.523847	1061.728775
ar	1298.871935	171.503182	1513.554546	994.643221

Table 5.2: Average of all cross-validation attempts for each model on location 7007405576125440 for daily data.

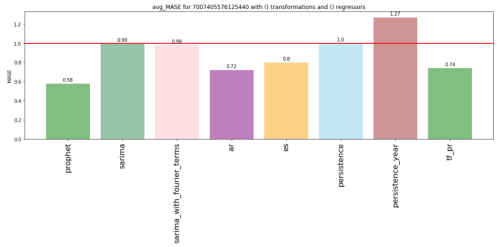


Figure 5.10: Average MASE of all cross-validation attempts for each model on location 7007405576125440 for the daily data.

seen that just for the location 7007405576125440 models seem to perform better than the Persistence model, being Prophet the best, performing almost twice as good as the Persistence model. In Figure 5.10 the values on Table 5.2 are represented in form of a bar chart for the particular location 7007405576125440 and the MASE metric.

Having seen that, now this location will be analyzed as we did for the hourly case and therefore all the different transformations and regressors will be evaluated. Keep in mind that for daily data just the holidays regressor has been implemented.

After applying the different data transformations one at a time we discover that applying Box-Cox before Prophet gives the best MASE, being 0.56 (Figure 5.11). Now we combine Box-Cox with the other data transformations in order to see if they show any improvement. It turns out that some of they do, being the combination of Box-Cox and Normalization the best of all of them. As it can be seen in Figure 5.12, we have improved the MASE metric from 0.58 to 0.52.

Forgetting for a moment about data transformation we use the Holiday regressor on the multivariate models: Prophet, SARIMA, SARIMA with fourier terms, and STS with TFP. Results in Figure 5.13 show that the multivariate models show an improvement. Particularly, Prophet gives a 0.52 MASE instead of 0.58 without holidays.

Finally, we combine the holidays with the combinations of transformations performing better and we finally got the best performance of a 0.49 MASE obtained with Prophet, Box-Cox transformation and holidays regressor. These results can be seen in Figure 5.14.

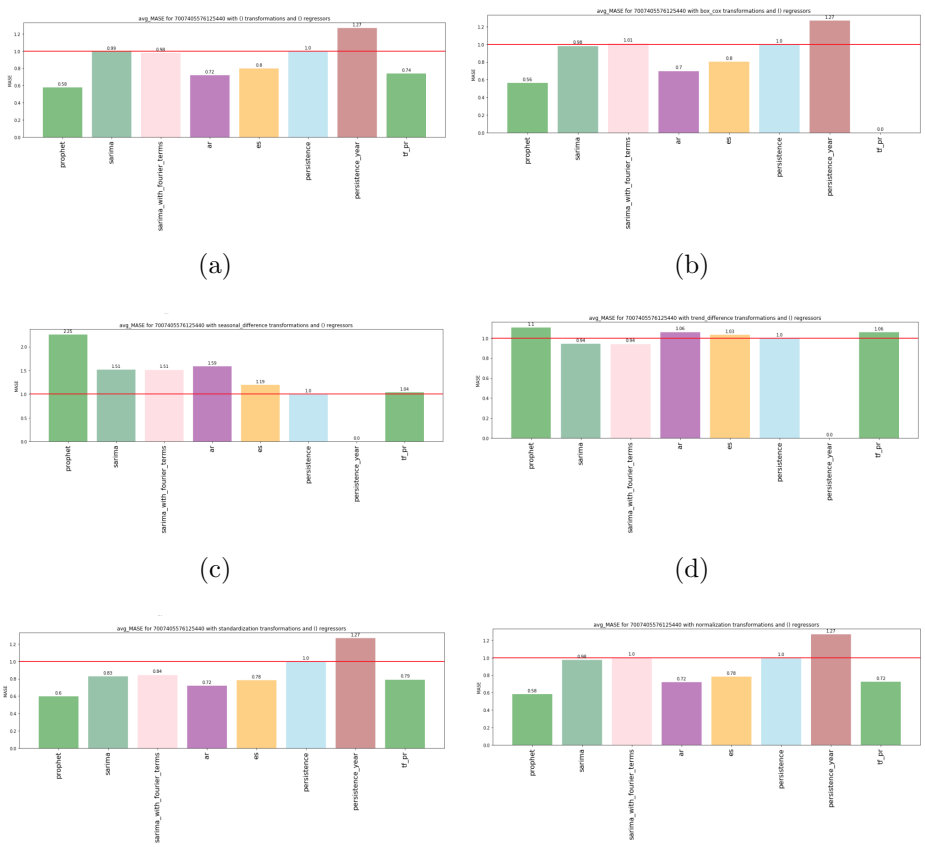
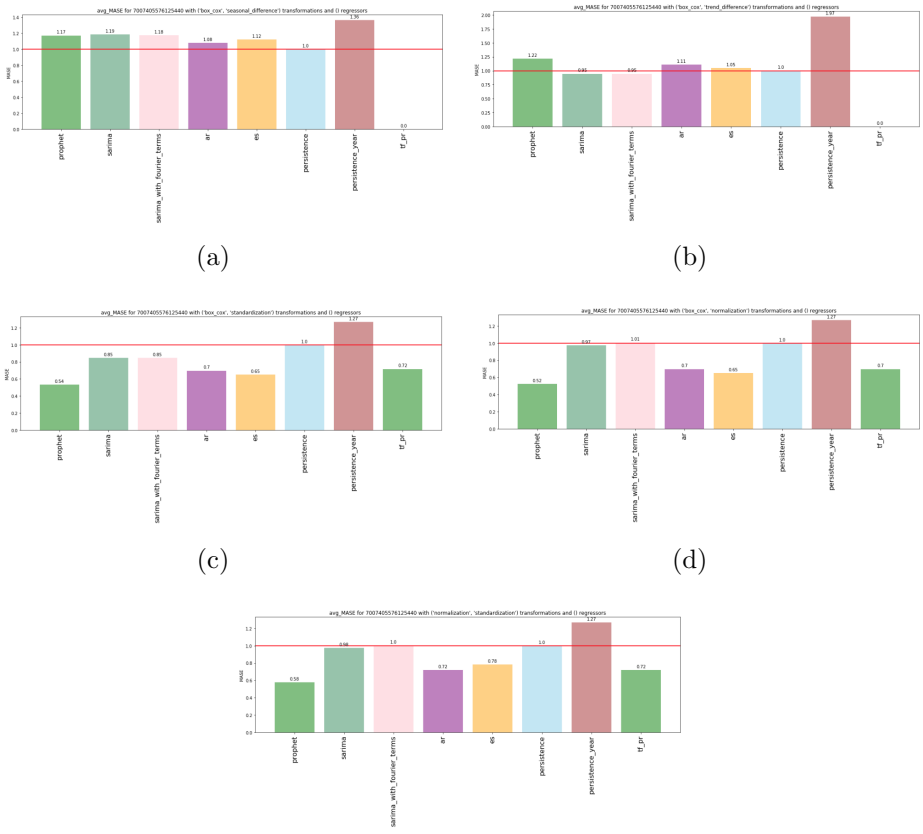


Figure 5.11: MASE metric for all the models and different data transformation for daily data on location 7007405576125440: (a) Without data transformations , (b) Box-Cox, (c) Seasonal difference, (d) Trend difference, (e) Standardization, (f) Normalization.

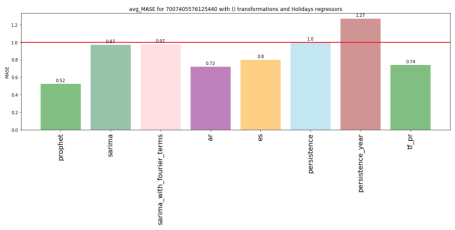


(a) (b)

(c) (d)

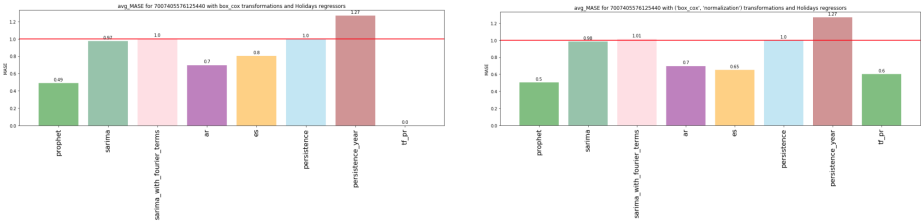
(e)

Figure 5.12: MASE metric for all the models and different data transformation combinations for daily data on location 7007405576125440: (a) WBox-Cox Seasonal difference , (b) Box-Cox Trend difference, (c) Box-Cox Standardization, (d) Box-Cox Normalization, (e) Box-Cox Standardization Normalization.

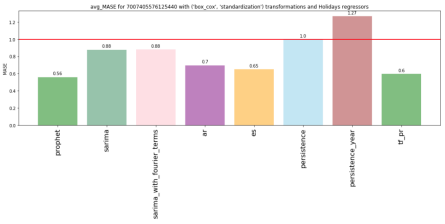


(a)

Figure 5.13: MASE metric for all the models without data transformation but using the holiday regressor on the multivariate models for daily data on location 7007405576125440.



(a) (b)



(c)

Figure 5.14: Combination of data transformation with regressors for daily data: (a) Box-Cox holidays, (b) Box-Cox and Normalization holidays, (c) Box-Cox and Standardization holidays.

Chapter 6

Discussion

In this section we discuss to what extend we have been able to answer the research questions asked at the beginning of the project. Remembering from the Introduction section we formulated three research questions:

- **Research Q1**: Is it possible to forecast future location occupancy using cell phone network data? If yes, how accurate are the forecasts obtained?
- **Research Q2**: Is there a single best model for the different locations?
- **Research Q3**: Are external variables useful in order to improve the performance of the models?

6.1 Applicability of location occupancy prediction

The most important question to solve in this thesis was if whether location occupancy prediction could be possible with the data provided and if it was the case, how accurate those forecasts could be. To measure the accuracy of our forecast we have used the main metrics for time series forecasting: MAE, MAPE and MASE, concentrating mainly on MASE. The first thing we have noticed is that the ability to forecast those signals really depends on each location we are trying to forecast. Some locations have more patterns in it than others and that leads to very different results. In this thesis we have analyzed all locations for all models and then we have focused on the location and models performing better. For both cases, hourly and daily, we have found that in general the location showing better performance was location 7007405576125440. The fact of location 7007405576125440 having

better predictions than the others can be given to that it seems to be the more stable of the three during the whole period, as can be appreciate in Figure 3.3 of section 3. In the hourly case, the best model for this location has been SARIMA obtaining a MASE of 0.99, meaning that our best model performs almost equally as the Persistence Weekly model. For the daily data, the best model on location 7007405576125440 has obtained a MASE of 0.49, meaning that our best model is twice times better than the Persistence Weekly model. To decide if the predictions are good enough will depend on the level of accuracy required for the application where the model will be used.

6.2 Best models for the prediction task

The conclusion after analysing the results obtained in this thesis is that there is not an existing unique model performing better in all the cases. Results highly depend on the particular signal to forecast. As an example, for the daily data, the best model is Prophet for location 700740557612544 and the Persistence Weekly for the other two locations.

Moreover, if we take a look to the different cross-validation attempts it can be seen that not always the same model is the best for all the cross-validation attempts in the same location.

Transformations and regressors also provide different performance on different models. For example, for the daily data, the best model is Prophet with the combination of Box-Cox transfromation and Holidays regressor but this same combination is not the one performing better for SARIMAX models.

6.3 Effect of using external variables

External variables are not always useful on improving the performance of the models. Surprisingly, some models can perform better with a particular regressor while others perform worse with the same regressor.

For example, for the hourly case none of the regressors help the models to increase performance while for the daily case, holidays regressor has helped to slightly improve performance of the majority of the models even though not in all of them.

Therefore, due to these results obtained, not compelling assertions can be done to confirm that any of the regressors really have a substantial impact

on predictions. Moreover, another fact to take into account, specially for the hourly case, is that some models take much more time when using regressors.

Chapter 7

Conclusions

This thesis has investigate and quantified the ability to predict the number of people in three different locations in UK for a certain point in time in the future. The data used in this endeavour are the past values of the data to predict as well as external regressors obtained during the same period of time. The data of the number of people for the different locations have been provided by the telecommunication company who obtained it aggregating the cell phone data of its users.

To achieve this, the developed models should be able to outperform the baseline model called Persistence model, which basically creates predictions copying the values of the signal to predict from one seasonality in the past. This project has implemented eight different time series models a part from the baseline model. Some of them are univariate and some of them are multivariate. Data transformation has been performed before fitting the models and also regressors signals have been included on the multivariate cases aiming to improve the final performance. In order to find the best combination of model, regressors, and data transformation, the time series data for the three locations have been evaluated with all the models and combinations of data transformations and regressors performing better.

Taking a look at the results obtained in the Results section, some facts about the capacity of the models can be stated. There is just one location, 7007405576125440, and just for the daily case, were the models perform clearly better than the Persistence Weekly model. Therefore, even though deeper analysis with more data would be helpful, the results obtained do not give a very promising view about forecasting these type of signals without any other external information. Thus, the first conclusion is that results really depend on each particular location. The three signals seem to follow different patterns and the only common aspect of all of them seems to be the daily and weekly seasonality pattern. In contrary, bigger seasonalities, like

yearly seasonality, do not seem to be reflected in any of them.

Models also have big differences depending on the slot of time to forecast and there is no consensus on a singular model performing better for the three different locations. Some transformations seem to improve some models in some cases while the only regressor that shows improvement is the holidays regressor and just for the daily case.

Making use of the error analysis tool created in this thesis we can see that cross-validation attempts with worst predictions show non-normal residual distributions for all the models meaning that they are not able to capture some patterns of the signal. Another big issue with the code built is that processing hourly data can take very long time even if we dispose of a good machine.

Finally, we believe that some future work can still be done in order to keep improving the project. We have been quite ambitious in this project and we have tackled all the most trendy models for time series in order to achieve the best possible results. Consequently, each model has only been analyzed in its surface and they have not been deeply tuned. Some future work around analyzing each model and tuning it in more detail should be done in order to extract more concise conclusions. The code implemented provides several tools explained in the Implementation section that can serve the analysts to understand better the signals in order to define better models. Bigger amount of computing resources will be needed in order to make this further tuning investigation doable since especially for the hourly case it takes very long time to run the models with cross-validation. One option to try in order to improve accuracy could be to perform ensemble of the models since it can be observed that not all the models are correlated when forecasting the different cross-validation attempts. Experimentation with multivariate predictions using different locations as variables could be worth to try, the VARMAX model seems to be able to do that. It would be also interesting to try different forecasting lengths since this project has just analyzed the one week case. And last but no least, due to the fact that signals show some strange peaks, some outlier detection techniques could be tried out which hopefully could improve the performance of the models.

Bibliography

[1] Nari Arunraj, Diane Ahrens, and Michael Fernandes. Application of SARIMAX model to forecast daily sales in food retail industry. *International Journal of Operations Research and Information Systems*, 7:1–21, Apr 2016.

[2] Herman J. Bierens. Armax model specification testing, with an application to unemployment in the Netherlands. *Journal of Econometrics*, 35(1):161–190, May 1987.

[3] George Edward Pelham Box and Gwilym Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day, Inc., San Francisco, CA, USA, 1990.

[4] George Edward Pelham Box and Gwilym Jenkins. *Time Series Analysis: Forecasting and Control*, volume 68. Jan 2016.

[5] Jason Brownlee. How to backtest machine learning models for time series forecasting, 2016. `https://machinelearningmastery.com/backtest-machine-learning-models-time-series-forecasting/`. Accessed 12 Sep 2019.

[6] Jason Brownlee. Time series forecasting with the long short-term memory network in python, 2017. `https://machinelearningmastery.com/time-series-forecasting-long-short-term-memory-network-python/`. Accessed 14 Jul 2019.

[7] Jacob Burnim Dave Moore and TFP Team. Structural time series modeling in tensorflow probability, 2019. `https://medium.com/tensorflow/structural-time-series-modeling-in-tensorflow-probability-344edac24083`. Accessed 15 Jul 2019.

[8] Google Developers. Structuraltimeseries, 2019. `https://www.tensorflow.org/probability/api_docs/python/tfp/sts/StructuralTimeSeries`. Accessed 18 Jul 2019.

[9] Google Developers. Tensorflow probability is a library for probabilistic reasoning and statistical analysis., 2019. `https://www.tensorflow.org/probability`. Accessed 12 Sep 2019.

[10] James Edbrooke. *Time Series Modelling Technique Analysis for Enterprise Stress Testing*. Thesis, Imperial College London, 2017.

[11] Robert Fildes, Andrew C. Harvey, Mike West, and Jeff Harrison. Forecasting, structural time series models and the kalman filter. *The Journal of the Operational Research Society*, 42:1031, Nov 1991.

[12] Stephanie Glen. Box cox transformation, 2016. `https://www.statisticshowto.datasciencecentral.com/box-cox-transformation/`. Accessed 12 Sep 2019.

[13] Robert Goodell Brown. Statistical forecasting for inventory control. *SERBIULA (sistema Librum 2.0)*, Jul 2019.

[14] Andrew Edward Harvey and Stanley Peters. Estimation procedures for structural time series models. 1990.

[15] T. J. Hastie and R. J. Tibshirani. *Generalized additive models*. London: Chapman & Hall, 1990.

[16] Charles C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 2004.

[17] Robin John Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, Australia, 2nd edition, 2018.

[18] Robin John Hyndman and Anne B Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, pages 679–688, 2006.

[19] Sean J. Taylor and Benjamin Letham. Forecasting at scale. *The American Statistician*, 72, Sep 2017.

[20] Richard H. Jones. Maximum likelihood fitting of arma models to time series with missing observations. *Technometrics*, 22(3):389–395, 1980.

[21] Numerical Optimization: Understanding L-BFGS. Limited-memory bfgs, 2014. `http://aria42.com/blog/2014/12/understanding-lbfgs`. Accessed 12 Sep 2019.

[22] Toppr Technologies Private Limited. Components of time series, 2019. https://www.toppr.com/guides/business-mathematics-and-statistics/time-series-analysis/components-of-time-series/. Accessed 05 Jul 2019.

[23] Spyros Makridakis and Michele Hibon. The m3-competition: Results, conclusions and implications. *International Journal of Forecasting*, 16:451–476, Oct 2000.

[24] Spyros Makridakis, Evangelos Spiliotis, and Vassilis Assimakopoulos. The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, Jun 2018.

[25] Spyros Makridakis, Evangelos Spiliotis, and Vassilis Assimakopoulos. Statistical and machine learning forecasting methods: Concerns and ways forward. *PLoS ONE*, 13, Mar 2018.

[26] Machine Learning Mastery. What is time series forecasting?, 2019. https://machinelearningmastery.com/time-series-forecasting/. Accessed 03 Jul 2019.

[27] Guy Nason. Time series introduction, 2019. http://www.people.maths.bris.ac.uk/~magpn/Research/LSTS/STSIntro. Accessed 03 Jul 2019.

[28] R. Nau. Summary of rules for identifying ARIMA models, 2019. https://robjhyndman.com/hyndsight/arimax/. Accessed 12 Jul 2019.

[29] Robert Nau. Summary of rules for identifying ARIMA models, 2013. https://people.duke.edu/~rnau/arimrule.htm. Accessed 14 Jul 2019.

[30] Christopher Olah. Understanding LSTM networks, 2015. https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed 15 Jul 2019.

[31] Alan Pankratz. *Forecasting with Dynamic Regression Models*. Wiley-Interscience, 1991.

[32] Paul S P Cowpertwait and Andrew Metcalfe. *Introductory Time Series With R*. Jan 2009.

[33] Steven L. Scott and Hal Varian. Predicting the present with bayesian structural time series. *International Journal of Mathematical Modelling and Numerical Optimisation*, 5:4–23, 2014.

[34] Jay Shah. Neural networks for beginners: Popular types and applications, 2019. https://blog.statsbot.co/neural-networks-for-beginners-d99f2235efca. Accessed 18 Jul 2019.

[35] Grzegorz Skorupa. Forecasting time series with multiple seasonalities using tbats in python, 2019. https://medium.com/intive-developers/forecasting-time-series-with-multiple-seasonalities-using-tbats-in-python-398a00ac0e8a. Accessed 14 Jul 2019.

[36] Urnasai and Gudikandula. Recurrent neural networks and lstm explained, 2019. https://medium.com/@purnasaigudikandula/recurrent-neural-networks-and-lstm-explained-7f51c7f6bbb9. Accessed 12 Sep 2019.

[37] Wikipedia. Autoregressive model, 2019. https://en.wikipedia.org/wiki/Autoregressive_model. Accessed 11 Jul 2019.

[38] Alina Zhang. How to build exponential smoothing models using python: Simple exponential smoothing, Holt, and Holt-Winters, 2019. https://medium.com/datadriveninvestor/how-to-build-exponential-smoothing-models-using-python-simple-exponential-smoothing-holt-and-da371189e1a1. Accessed 11 Jul 2019.

Appendix A

Backshift notation

Backshift notation is an easy and useful way to define time series models with lags. B is the backshift operator and is used to represent a delay on a signal. In other words, B, operating on y_t, has the effect of shifting the data back one period.

$$By_t = y_{t-1} \,. \tag{A.1}$$

And consequently, two applications of B to y_t shifts the data back two periods:

$$B(By_t) = B^2 y_t = y_{t-2} \,. \tag{A.2}$$

Backward shift operator is convenient for describing the process of differencing. A difference between a signal and itself lagged one period would be represented by $(1 - B)$:

$$y'_t = y_t - y_{t-1} = y_t - By_t = (1 - B)y_t \,. \tag{A.3}$$

Similarly, a second order difference would be represented as:

$$y''_t = y_t - 2y_{t-1} + y_{t-2} = (1 - 2B + B^2)y_t = (1 - B)^2 y_t \,. \tag{A.4}$$

Therefore, extrapolating we obtain htat the dth-order difference can be written as

$$(1 - B)^d y_t. \tag{A.5}$$

Backsfhift nottation becomes very useful when combining differences, as the operator can be treated using ordinary algebraic rules. For example, a seasonal difference followed by a first difference can be written as

$$\begin{aligned}(1 - B)(1 - B^m)y_t &= (1 - B - B^m + B^{m+1})y_t \\ &= y_t - y_{t-1} - y_{t-m} + y_{t-m-1}.\end{aligned} \tag{A.6}$$